Marquette University
e-Publications@Marquette

Dissertations (2009 -) | Dissertations, Theses, and Professional Projects

Automating the Analysis of Uncertainties in Multi-Body Dynamic Systems Using Polynomial Chaos Theory

Paul Ryan
Marquette University

Recommended Citation
Ryan, Paul, "Automating the Analysis of Uncertainties in Multi-Body Dynamic Systems Using Polynomial Chaos Theory" (2018). *Dissertations (2009 -)*. 835.
https://epublications.marquette.edu/dissertations_mu/835

AUTOMATING THE ANALYSIS OF UNCERTAINTIES IN MULTI-BODY DYNAMIC SYSTEMS USING POLYNOMIAL CHAOS THEORY

by

Paul Ryan, M.S.

A Dissertation Submitted to the Faculty of the
Graduate School, Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy

Milwaukee, Wisconsin

December 2018

ABSTRACT

AUTOMATING THE ANALYSIS OF UNCERTAINTIES IN MULTI-BODY DYNAMIC SYSTEMS USING POLYNOMIAL CHAOS THEORY

Paul Ryan, M.S.

Marquette University, 2018

Variation occurs in many multi-body dynamic (MBD) systems in the geometry, mass, or forces. This variation creates uncertainty in the responses of an MBD system. Understanding how MBD systems respond to the variation is imperative for the design of a robust system. However, the simulation of how variation propagates into the solution is complicated as most MBD systems cannot be simplified into to a system of ordinary differential equations (ODE). This dissertation derives and automates the uncertainty analysis of an MBD system with variation. The first step to automating the solution is to create a robust algorithm based on the Constrained Lagrangian formulation for deriving the equations of motion. Using the Constrained Lagrangian algorithm as a starting point, the new process presented uses polynomial chaos theory (PCT) to embed the stochastic parameters into the equations of motion. To accomplish this, the concept of Variational Work is derived and implemented in the solution. Variational Work applies PCT to the energy terms and Principle of Virtual Work of the Constrained Lagrangian rather than applying PCT on the equations of motion. Using an automated process for applying PCT to an MBD system, four example problems are solved. Each of these problems is compared to a Monte Carlo analysis using the deterministic automation process. Three of the examples are non-textbook based problems, which show limitations in the application of PCT to an MBD system. The limitations and the possible solutions to overcoming them are discussed.

ACKNOWLEDGMENTS

Paul Ryan, M.S.

I would like to express my appreciation and thanks to my entire committee for providing guidance throughout my research and for challenging me when I needed challenging. Special thanks to my advisor, Dr. Philip Voglewede, for his support, his advice, and especially his help in finding solutions where I was certain there were none to be found.

Thank you to my former colleagues, Dr. Sree Thampi, Dr. Robert Clark and Dr. Ian Fialho, who showed by example the impact a doctorate could have in industry. They pushed me to pursue my doctorate and provided invaluable guidance in the beginning of my career.

Thank you to my parents, Thomas and Kathryn Ryan, for their love and instilling in me early on the importance of a good education. Thank you to my father-in-law and my mother-in-law, Ronald and Paula Washko, for their unwavering encouragement in pursuit of my doctorate.

Thank you to my two children, Julia and Nicholas, for their love and patience while I have been away doing my research. They are my strength and motivation, not only during this process, but in all parts of my life. Most importantly, I would like to thank my wife, Carol Ann, without whom none of this would have been possible. Thank you for your unwavering support, your sacrifice, and for believing me when I said I would finish this someday.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

Syntax

$\underline{f}$	Vector
$\mathbf{F}$	Matrix
$\mathring{f}$	Variable has been projected onto orthogonal polynomial basis
$\dot{f}$	First time derivative with respect to time
$\ddot{f}$	Second time derivative with respect to time
$\langle f, g \rangle$	Inner product
$\underline{f} \cdot \underline{g}$	Dot product

Multi-Body Dynamics

$\mathbf{0}$	Matrix of zeros
$\mathbf{A}_i$	Rotation matrix from the origin to body *i*
$\mathbf{A}'_i$	Rotation matrix from body *i* to another frame on body *i*
$\underline{C}\left(\underline{\dot{q}}, \underline{q}, t\right), \underline{C}$	Nonlinear acceleration inertial terms
$\delta\underline{r}_i$	Virtual displacement of force *i*
$\delta\mathring{\underline{r}}_i$	Variational virtual displacement of force *i*
$\delta\underline{\theta}_i$	Virtual displacement of moment *i*
$\delta\mathring{\underline{\theta}}_i$	Variational virtual displacement of moment *i*
$\delta\mathrm{Work}$	Virtual Work
$\delta\mathring{\mathrm{W}}\mathrm{ork}$	Variational Virtual Work
$\underline{e}$	Quaternion vector
$\underline{F}$	Force Vector
$\underline{F}_i$	Force *i*
$\mathring{\underline{F}}_i$	Variational force *i*
$\mathbf{G}_i$	Matrix transformation from quaternion velocities to body fixed angular velocity *i*
$\underline{g}$	Gravity vector
$\underline{\gamma}$	Vector of nonlinear acceleration constraint terms
$\mathbf{I}$	Identity matrix
$\mathbf{J}_i$	Inertia tensor of body *i*
$\mathbf{J}_p$	System Jacobian with respect to the system states
$\mathbf{J}_{\mathrm{sys}}$	System Jacobian

$\mathbf{J}_v$	System Jacobian with respect to the time derivatives of the system states
$\underline{\lambda}$	Lagrange multipliers
$\underline{\mathring{\lambda}}$	Variational Lagrange multipliers
$\mathbf{M}\left(\underline{q}\right), \mathbf{M}$	Inertia matrix
m_i	Mass of body *i*
$\mathbf{M}_i$	Mass matrix of body *i*
$\underline{M}_i$	Moment *i*
$\underline{\mathring{M}}_i$	Variational moment *i*
$\tilde{\mathbf{\Omega}}$	Skew-symmetric matrix for cross-product in 2D
$\underline{\omega}_i$	Angular velocity of body *i*
$\underline{\Phi}\left(\underline{q}, t\right), \underline{\Phi}$	Vector of constraint equations
$\underline{\mathring{\Phi}}$	Vector of variational constraint equations
$\Phi_{[\text{name}]}$	Constraint or joint "name"
$\mathbf{\Phi}_q\left(\underline{q}, t\right), \mathbf{\Phi}_q$	Constraint Jacobian
$\mathring{\mathbf{\Phi}}_q$	Variational constraint Jacobian
$\mathbf{\Phi}_{q_i}$	The *ith* row of the constraint Jacobian
$\underline{\Phi}_t$	Vector of the position constraints explicitly differentiated with respect to time
$\underline{\mathring{\Phi}}_t$	Variational vector of the position constraints explicitly differentiated with respect to time
$\underline{\Phi}_{tt}$	Vector of the position constraints explicitly differentiated with respect to time twice
$\underline{Q}\left(\underline{\dot{q}}, \underline{q}, t\right), \underline{Q}$	Vector of generalized forces
$\underline{\mathring{Q}}$	Variational vector of generalized forces
$\underline{q}$	Generalized coordinates
Q_i	Generalized force of generalized coordinate *i*
$\mathring{Q}_i$	Variational generalized force of variational generalized coordinate *i*
$\underline{q}_i$	Generalized coordinates for body *i*
q_i	Generalized coordinates *i*
$\mathring{q}_i$	Variational generalized coordinates *i*
$\underline{q}_{Ti}$	Translational generalized coordinates for body *i*
$R\left(\text{axis}, \text{angle}\right)$	Rotation about an "axis" by an "angle"
$R_{2D}\left(\text{angle}\right)$	A 2D rotation by an "angle"
$\underline{r}_i$	Point from the origin to the origin of body *i*
$\underline{s}'_i$	Point from the origin of body *i* to a point on body i
T	System kinetic energy
$\mathring{T}$	System variational kinetic energy
V	System potential energy
$\mathring{V}$	System variational potential energy
V_{elastic}	System elastic potential energy

$\overset{\circ}{V}_{\text{elastic}}$	System variational elastic potential energy
V_{grav}	System gravitational potential energy
$\overset{\circ}{V}_{\text{grav}}$	System variational gravitational potential energy
$\underline{x}_p$	Vector of position states for the system
$\underline{x}_v$	Vector of velocity states for the system
$\underline{x}_\lambda$	Vector of Lagrange multiplier states for the system
$\underline{x}_\mu$	Vector of additional Lagrange multiplier states for the system

Polynomial Chaos Theory

Ψ_i	Polynomial of selected basis
$\underline{\xi}$	Vector of Random variables
$W\left(\underline{\xi}\right), W\left(\xi\right)$	Weighting function of basis
X, Y, Z	A second order random process with finite variance

CHAPTER 1

Uncertainty Analysis of a Multi-Body Dynamics System

Variation occurs in every system ever designed, built or analyzed. The impact that this variation has on the system needs to be understood to ensure the system stays in its operational limits or design envelope. Not knowing how the variation affects the loads in the system can lead to an undersized part that breaks or an oversized part that may cause added weight for non-optimal performance. Characterization of the uncertainties allows for a more robust design and improved performance. Quantifying the uncertainties of system responses is critical to the design or analysis of these systems.

Including dynamics in the design and analysis of a system adds its own unique challenges. The derivation of the governing equations of a dynamic system with only a handful of bodies moving through three-dimensional space quickly becomes overwhelming to do by hand. As the number of bodies of the dynamic system increases, the difficulty in deriving the equations increases. One solution for simplifying the derivation of the equations of a dynamic system consisting of a large number of bodies is using a Multi-Body Dynamics (MBD) method, such as one from Haug [22] or using a commercial of-the-shelf (COTS) MBD program such as MSC.Adams.

The most popular way to handle variation in an MBD system is using a Monte Carlo (MC) analysis. Based on the variation of the inputs or parameters, a set of random

samples is simulated using the MBD model. Depending on the number of inputs or parameters with variation, in conjunction with how large and complex the MBD model is, an MC analysis can have a very long cycle time. A very long cycle time can translate into days, weeks, or more of analysis, making the analysis itself unrealistic. Therefore, there is a need for performing an uncertainty analysis (UA) on an MBD system which eliminates long cycle times, while not sacrificing accuracy.

To achieve a replacement method for a UA in an MBD system that eliminates long cycle times while maintaining accuracy, the goal of this dissertation is to have the UA set-up and simulated using a robust computer algorithm. To make this possible, additional criteria are needed.

- First, the method needs to be general and not an ad hoc approach. The new method needs to be applied with no additional knowledge beyond that it is an MBD system. Regardless of user, the same equations of motion are arrived at; it is repeatable and unambiguous.
- Second, there needs to be a certain level of automation. Given any MBD system, the resulting method requires little to no user inputs regarding the structure or coordinates used. Because MBD systems can be very large, the equations of motion cannot be built manually. Manually building a large system takes a lot of time and effort, and opens the system to many sources of error.

- Third, the method needs to be at least as fast as an MC analysis. If a system needs n MC simulations to characterize the system response correctly, then the method needs to be as quick or quicker than running those n simulations serially. In theory, parallel computing can reduce the time to perform an MC analysis. Unfortunately, there are many of variables that cannot be controlled such as hardware specifications, licensing, etc. Therefore, for objectivity, serial simulations is used as the metric.
- Fourth, post-processing of the results needs to be quick and easy. There should not be a lot of time associated with trying to quantify the uncertainty of the responses. MBD system responses are time histories with at least a few hundred time steps in the response. If there are a large number of random variables in the system, post-processing through time can add significantly to the cycle time.

The goal is to automate the UA of the MBD system using a method that meets the four above criteria. Ensuring these criteria are met, a robust analysis method can be achieved. To accomplish this, an MBD system needs to be defined followed by the definition of a UA.

In this chapter, an MBD system is defined. Next, a UA is defined. Finally, the application of a UA to an MBD system is explained.

1.1 Multi-Body Dynamic System

To automate the analysis of the MBD system with uncertainty, first an MBD system needs to be defined. An MBD system is a system of rigid or flexible bodies with kinematic constraints relating them, as well as forces and moments acting on the system. An MBD system can be represented in its most compact form, meaning the fewest equations, by a system of ordinary differential equations (ODEs) that equal the number of degrees of freedom (DOF) of the system. The largest set of equations occur when each body is represented by a set of ODEs equal to the number of unconstrained DOF in that space, plus the kinematic constraint equations connecting the bodies. This model is referred to as a maximal set of generalized coordinates [3] or a maximal set of differential algebraic equations (DAE). Therefore, there is an ODE for each generalized coordinate in the system and a set of algebraic equations equal to the number of kinematic constraint equations. For an open kinematic chain, the system usually is expressed as a system of ODEs, while a closed kinematic chain system usually is not. Although it may be possible represent the closed kinematic chain system as a system of ODEs, the cleanest or most compact form is a system of DAEs. In general, an MBD system consists of both open and closed kinematic chains. For this dissertation, only rigid bodies and holonomic constraints are considered because most MBD systems contain mostly rigid bodies with holonomic

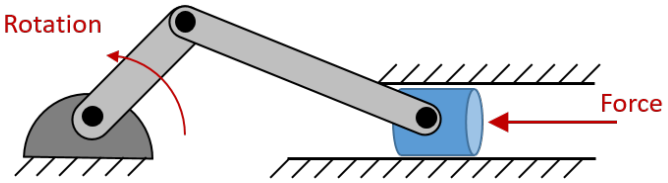

Figure 1.1: Piston or slider-crank mechanism

constraints. Most computer-aided engineering MBD software focuses on holonomic constraints as the basis of the system [22, 38].

A commonly used example of an MBD system is a piston connected to a crank-shaft or a slider-crank, as seen in Fig. 1.1. The mechanism contains three bodies or parts that are connected by three revolute (or pin) joints and one prismatic (or translational) joint. The system is a one DOF system; if the piston head moves, the crank rotates, and vice versa. Since the system has one DOF, it can be modeled in its most compact form as one ODE. Assuming this is a planar problem and each part is represented with three degrees of freedom when unconstrained, then the maximal DAE are nine ODEs with eight algebraic constraint equations. There are two algebraic constraint equations for each revolute joint and two for the translational joint.

MBD systems can get very large, and it is not easy to represent them as a system of ODEs. Systems can have well over a hundred bodies with hundreds of DOFs. For instance, an MBD system can be a tracked vehicle, such as a tank in Fig. 1.2 . If a propulsion MBD model is needed, then each of the track links, road wheels, drive

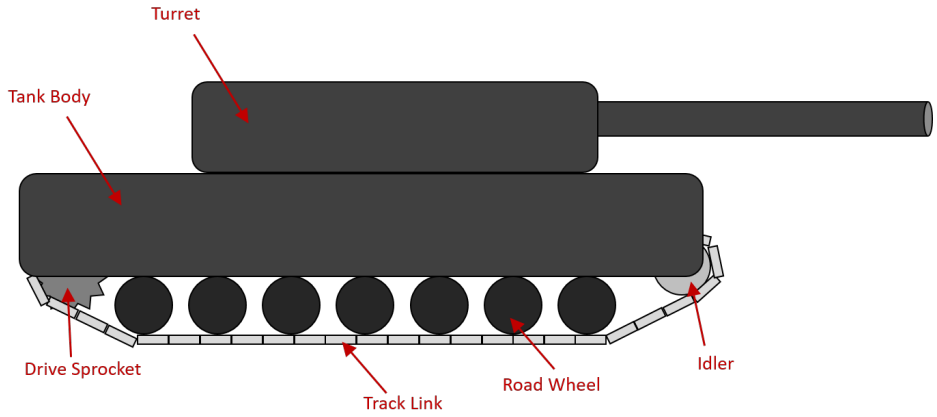

Figure 1.2: Tank propulsion model

sprocket, etc., need to be modeled. The system is a couple hundred generalized coordinates with at least a hundred constraint equations. Also, there is hundreds of contact forces created to capture the reaction between the track links and the road wheels. Thus, to model a tank with uncertainty, an automated approach building the equations of motion that reduces cycle time while maintaining a certain level of accuracy is required. Typically, an MBD system can have a large number of rigid bodies, which means the system contains tens or hundreds of generalized coordinates with a large number of kinematic constraints. Adding to the complexity of the system are nonlinearities. These occur in the kinematic constraint equations because the system is defined in SE(3) space [39], which based upon the generalized coordinates chosen results in many trigonometric (e.g., $\sin$ and $\cos$) functions. Also, nonlinearities can come from the kinetics of the problem. Conservative forces can occur in the gravitational potential or in the

elastic potential. For the non-conservative forces, nonlinearities can occur in the losses, such as friction or damping, in contact or in generic forcing functions.

Due to the size of the MBD system, a robust consistent method is needed to ensure consistent results. To apply a robust consistent method, automation is needed to derive the equations of motion and solving the system. Finally, the size of the MBD system creates a large set of output data. A post-processing method is needed to view the results requested by the user.

1.2 Uncertainty Analysis

The next step is to define exactly what is a UA. A UA assesses the system outputs as the errors in the inputs or parameters propagate through the system [7]. In other words, it is a method for quantifying the uncertainty in system outputs [46]. Typically, the sources for errors are described statistically, using parameters such as mean and standard deviation for a normal distribution for example [47]. The purpose of the UA is to determine the probability density function (PDF) of the response [44]. A UA is different from a sensitivity analysis (SA), which Saltelli [47] describes as a "study of how the uncertainty in the output of a model (numerical or otherwise) can be apportioned to different sources of uncertainty in the model input." A UA quantifies the probability of the system outputs, while an SA, using only extreme values of the uncertainties, determines

how the outputs are influenced by the uncertainties [28]. Ideally, both a UA and an SA are performed in all MBD analyses.

In the case of MBD systems, uncertainty plays a big role and needs to be understood. Whether the uncertainty comes from the machining tolerances, material changes, or noise from the input, understanding the response of the system due to these variations is critical. For example, these uncertainties influence the forces and moments acting on certain parts of the system. Not taking into account how the variation affects the loads in the system can lead to an undersized part that fails or an oversized part that may cause added weight. The added weight adds to increased cost of material and more expensive, oversized actuators. Characterization of the uncertainties allows for a more robust design and improved performance. Also, if an SA also is performed, knowing how variation influences the system responses allows for more strategic testing and analysis. Only parameters which are important to a particular response need to be measured or modeled.

1.2.1 Sources of Uncertainty in an MBD System

In an MBD system, there are many areas where uncertainties can enter. However, uncertainties can be classified into three major categories. The first includes geometric uncertainties, or those uncertainties that affect the physical shape of a body. These can affect the volume of the body, which affects the mass or inertia. Also, they affect the

kinematic constraints and the constraint Jacobian. Finally, the geometric uncertainties can impact the application of the generalized forces.

The second category of uncertainty includes material properties. The material property of interest for a rigid body system is the density. For flexible bodies, the modulus of elasticity, the shear modulus, and Poisson's ratio also are important. The density affects the mass or inertia and the other material properties affect the stiffness matrix of a flexible body. This dissertation is only focusing on rigid body systems, and flexible bodies are included for completeness in this description.

The last category is kinetic uncertainty. Kinetic uncertainty is a broader category because it includes anything that affects the forces and moments acting on the system, which ultimately affect the generalized forces. Examples include the stiffness of a spring, the coefficient of kinetic friction, contact forces, or a generic forcing function. It is especially true, because of this last category of uncertainty, that a general formulation to dealing with uncertainties is needed. Since there is no restriction on the type of forces or moments that are applied to an MBD system, a general approach that allows the generalized forces to be computed automatically is critical.

1.3 Dissertation Outline

This dissertation is organized as follows. In Chapter 2, the Literature Review, various UA's are explained and evaluated as possible solutions to a UA of an MBD system. Chapter 3 explains an MBD system from the derivation method of Constrained Lagrangian to numerical simulation of the DAE. A slider-crank is used as an example to show how the MBD method is applied. In Chapter 4, the concept of Variational Work is derived and applied to the Constrained Lagrangian. The slider-crank is used again to demonstrate the application of Variational Work. Chapter 5 compares the slider-crank formulations and three real-world examples: a bouncing ball, a motorcycle and spherical mechanism. Finally, Chapter 6 contains the forward work and conclusions.

CHAPTER 2

Literature Review of Uncertainty Analyses Their Application to Multi-Body Dynamic Systems

There are many commercial of-the-shelf (COTS) Multi-Body Dynamics (MBD) programs widely used in industry and academia. Each of these programs has advantages and disadvantages when it comes to solving the MBD problem. Also, there are various UA methods that can be used to analyze uncertainties in an MBD system. Each of these UA methods is presented and evaluated in the context of the MBD formulation.

COTS MBD programs include (not an exhaustive list) MSC.Adams [36], Virtual.Lab Motion [51], Simulia [8], RecurDyn [35] and Mathworks' Simscape Multibody [32]. These programs are used extensively to solve complex problems, with degrees of freedom (DOF) in the hundreds. The software packages all build their version of the maximal set of equations [22, 38]. However, each of these codes have a rigid formulation for how many generalized coordinates each part can have. For example, MSC.Adams defines all rigid bodies with six generalized coordinates, three translational and three Euler angles (ZXZ), which cannot be changed [38]. This rigid formulation prevents the use of some of the UA techniques that do not use a deterministic model.

For decades, many people have tried to solve the UA problem. Some of the major classifications of UA methods are sampling techniques, moment equations, perturbation

theory, remote surface method (RSM), and polynomial chaos theory (PCT). Each of these methods has advantages and disadvantages when applied to an MBD system. Each of these methods in this chapter is organized with sections of "General Approach" and "Assessment."

2.1 Sampling Techniques

The following section discusses sampling techniques and some common uses of them. Sampling Techniques are evaluated for use in an MBD system.

2.1.1 General Approach

Sampling techniques are statistical approaches that take samples from a population of inputs and use these to predict the population of the output. The more samples of the input population, the more accurate the prediction of the output population is. The hope is that there are enough samples to represent sufficiently the whole population [12]. Unfortunately, the choice of sampling technique influences the error in the prediction of the probabilistic nature of the system outputs [59]. There are two types of sampling techniques: probability sampling and nonprobability sampling. The probability sampling bases its sampling from probability density functions (PDF), and these fall into the family of Monte Carlo (MC) methods [12]. Nonprobability sampling does not involve randomness and is outside the scope of this dissertation [12].

MC methods are the primary technique for performing an uncertainty analysis. According to Halton [19], “the Monte Carlo method is generally defined as representing the solution of a problem as a parameter of a hypothetical population, and using a random sequence of numbers to construct a sample of the population, from which statistical estimates of the parameter can be obtained.” In other words, independent realizations of the random inputs or parameters are created from their respective PDFs, then the set of realizations is run through a deterministic model, and the realizations of the outputs are collected [61]. Because the independent realizations typically are generated by a computer, using a deterministic algorithm to create a random number, they are technically pseudorandom [12].

There are many MC methods, each one has advantages and disadvantages. However, they all follow a simple process [21]:

1. Determine the PDFs for the random input variables,
2. Repeatedly sample the PDFs to form a simulation set,
3. Simulate the deterministic model with each set, and
4. Collect response data and compute statistical information.

Two popular MC methods that are discussed in the following sections are the brute force and Latin Hypercube methods.

2.1.2 Brute Force Monte Carlo

The simplest MC method is the brute force method. Each variational input or parameter is sampled n times based on its PDF. For the purpose of this dissertation, the PDF is assumed to be known. A set of all the random variables is created, placed into the deterministic model, and simulated. The deterministic model is evaluated or simulated n times. The responses of each of the simulations is collected, and statistical information are extracted, typically the mean and standard deviation [61].

2.1.3 Latin Hypercube Monte Carlo

Latin Hypercube MC tries to reduce the number of samples that are needed to define a system response accurately in the brute force method [12]. Each random variable is divided into equal non-overlapping probability intervals [12]. A single random sample from each bin is then computed [12]. All the samples of the random variables are paired randomly to form an MC set [12]. Each of these sets is then simulated using the deterministic model, similar to the brute force method. Statistical information then is extracted from all of the system responses. Since the Latin Hypercube method represents the distributions more efficiently, it takes fewer samples to achieve an accurate mean and standard deviation estimate [25].

2.1.4 Assessment

The MC methods can be applied in a general sense to any MBD problem. This is primarily why it is one of the most commonly used methods to quantify uncertainty in an MBD system. It can be combined easily with existing MBD software such as MSC.Adams or any MBD system building method, as from [22]. Unfortunately, since MBD systems are large and complex models, it is not sufficiently fast. Also, since MBD systems are simulated through time, the statistical information of each response is computed at every time step.

2.2 Moment Equations

Another type of UA uses moment equations. Moment equations are discussed in the following section. This type of UA is then evaluated for use in an MBD system.

2.2.1 General Approach

Moment equations are an uncertainty analysis that computes the moments of the responses directly using a Taylor series expansion about a value [61]. The first moment is the mean, and the second moment is the variance or the standard deviation squared. Typically, the mean value is used as the point to expand the Taylor series [58]. Given an MBD DAE system, if each equation is designated $G_k\left(\underline{x}\right)$, where $\underline{x}$ is a vector of n random

variables, then the Taylor series expansion is [44]

$$\begin{aligned} G_k\left(\underline{x}\right) =& G_k\left(\underline{\mu}\right) \\ &+ \sum_{i=1}^{n} \frac{\partial G_k\left(\underline{x}\right)}{\partial x_i}\Big|_{\underline{x}=\underline{\mu}} \left(x_i - \mu_i\right) \\ &+ \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 G_k\left(\underline{x}\right)}{\partial x_i \partial x_j}\Big|_{\underline{x}=\underline{\mu}} \left(x_i - \mu_i\right)\left(x_j - \mu_j\right) \\ &+ H.O.T. , \end{aligned} \tag{2.1}$$

where $H.O.T.$ is short for higher order terms. There are generally two methods for calculating the mean and variance of the system, the first order second moment (FOSM) and the second order second moment method (SOSM). In [29], a detailed derivation is shown for how the mean and variance are calculated. Assuming the random variables are independent, to calculate the first moment, the mean, and the second moment, the variance [58],

$$\mu_{G_k} = G_k\left(\underline{\mu}\right) \ , \text{ and} \tag{2.2}$$

$$\sigma^2_{G_k} = \sum_{i=1}^{n} \left(\frac{\partial G_k\left(\underline{x}\right)}{\partial x_i}\Big|_{\underline{x}=\underline{\mu}} \right)^2 \sigma^2_{x_i} \ , \tag{2.3}$$

where σ_{x_i} is the standard deviation of the random variable. The first and second moments for the SOSM formulation are

$$\mu_{G_k} = G_k\left(\underline{\mu}\right) + \frac{1}{2}\sum_{i=1}^{n} \frac{\partial^2 G_k\left(\underline{x}\right)}{\partial x_i^2}\Big|_{\underline{x}=\underline{\mu}} \sigma_{x_i}^2 \text{ , and} \tag{2.4}$$

$$\sigma_{G_k} = \left(\frac{\partial G_k\left(\underline{x}\right)}{\partial x_i}\Big|_{\underline{x}=\underline{\mu}}\right)^2 \sigma_{x_i}^2 + \frac{1}{2}\left(\frac{\partial^2 G_k\left(\underline{x}\right)}{\partial x_i^2}\Big|_{\underline{x}=\underline{\mu}} \sigma_{x_i}\right)^2. \tag{2.5}$$

Each equation in the system is evaluated using either the FOSM or the SOSM equations. The new system is twice as large as the original system, one system for the mean and one system for the variance. The system then can be integrated or simulated with respect to time to get the means and variances of the responses.

2.2.2 Assessment

Depending on the size of the MBD system, it might be impractical to build the system governing equations analytically. The impracticality is true regardless of whether it is a minimal DOF system consisting of only ODEs or a maximal system DAE. The first and/or second partial derivatives of the system equations with respect to the random variables need to be found, and they need to be squared. This is easier for the FOSM than it is the SOSM, but the SOSM in theory is more accurate [20].

In general, the moment equation formulations fail when applied to highly nonlinear systems [30]. In the case of the MBD system, except in a very rare case, the

system is nonlinear with polynomial, trigonometric, exponential, etc., terms. Also, moment equations are not able to handle discontinuities, such as contact or friction, which cannot be linearized.

To get the FOSM or SOSM to work, the user needs to be involved in almost every step of the way making simplifications. Applying Moment Equations to an MBD system is very ad hoc. This violates criteria one and two in Chapter 1, and therefore Moment Equations are not an appropriate method to use with a general MBD system.

2.3 Perturbation Method

A third type of UA is the perturbation method. The perturbation method is discussed in the following section. This type of UA then is evaluated for use in an MBD system.

2.3.1 General Approach

Perturbation theory sometimes is confused with moment equation methods in the literature. There are similarities which involve Taylor series expansions, but in the perturbation method, the components of the governing equations are expanded via Taylor series, including the responses, about the mean of the random variables. In the moment equation method, the moments are computed directly from the governing equations [61].

The coefficients of the expansions of the responses are calculated using a closed-form solution [53]. This method is used most commonly in structural dynamics. All of the examples that were found [11, 31, 34, 53, 54] were of the static finite element analysis (FEA) form

$$\mathbf{K}\left(\underline{\alpha}\right)\underline{x}\left(\underline{\alpha}\right)=\underline{F}\left(\underline{\alpha}\right)\ , \tag{2.6}$$

where $\mathbf{K}$ is the stiffness matrix, $\underline{F}$ is the applied force vector, $\underline{x}$ is the displacement vector and $\underline{\alpha}$ is a vector of random variables. In [54], the example problems look at variation of the system response, $\underline{x}$, about the equilibrium for the system. Each term is expanded about its mean via a Taylor series expansion. For example, the stiffness is expanded

$$\begin{aligned}\mathbf{K}\left(\underline{\alpha}\right)=&\mathbf{K}\left(\underline{\mu}\right)\\&+\sum_{i=1}^{n}\frac{\partial\mathbf{K}\left(\underline{\alpha}\right)}{\partial x_i}\Big|_{\underline{\alpha}=\underline{\mu}}\left(\alpha_i-\mu_i\right)\\&+\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial^2\mathbf{K}\left(\underline{\alpha}\right)}{\partial\alpha_i\partial\alpha_j}\Big|_{\underline{\alpha}=\underline{\mu}}\left(\alpha_i-\mu_i\right)\left(\alpha_j-\mu_j\right)\\&+H.O.T.\end{aligned} \tag{2.7}$$

Each term in Eqn. (2.6) is expanded like Eqn. (2.7) using the same Taylor series expansion order and substituted back into Eqn. (2.6). Like terms are grouped on both sides of the equation based on similar order coefficients of α_i [54]. If possible, then $\underline{x}$ is solved for from each of theses terms. Once the coefficients of the Taylor series expanded response are found, the mean and variance can than be computed. Using a second order Taylor

series expansion, a two degree of freedom spring system had a very accurate approximation of the real response [34].

2.3.2 Assessment

All of the solutions found using the perturbation method were static FEA problems. It is difficult to determine how to implement this method for an MBD system where there are derivatives of the responses. In the simple case of

$$\mathbf{M}\underline{\ddot{x}} + \mathbf{K}\underline{x} = \underline{F}\,, \tag{2.8}$$

where $\mathbf{M}$ is the mass matrix, the second derivative of $\underline{x}$ can be expanded by taking the second derivative of the Taylor series response. The system then can be simulated through time to determine the coefficients of the Taylor series expansion. The method seems to be very arduous for a very large system, where taking partial derivatives for the Taylor series can become unrealistic and increase computational time [34]. Also, in general, using a second order or higher Taylor series expansion approximation only works for Gaussian or normal distributions [53]. Finally, for perturbation methods to be accurate, the inputs and outputs magnitude of uncertainties need to be under 10% [61]. This method also appears to be an ad hoc method; the user is involved making decisions at every step of the process. Due to the process seeming more manual, it is not easily automated. Since the perturbation method appears unrealistic to implement for a large system, in violation of

criteria one and two in Chapter 1, the perturbation method is not an appropriate method to use with a general MBD system.

2.4 Response Surface Method

A fourth type of uncertainty analysis is the response surface method. This method is discussed in the following section. The response surface method then is evaluated for use in an MBD system.

2.4.1 General Approach

There are many uses of the term "response" surface in the literature. Myers's [40] definition is accepted for this dissertation, "a collection of tools in design or data analysis that enhance the exploration of a region of design variables in one or more responses." In general, there are four steps for the response surface method (RSM) [13]:

1. Sensitivity analysis to determine important inputs or parameters,
2. Determine a simplified model that represents the original model,
3. Calculate moments of the RSM model, and
4. Fit a statistical model to the moments.

The statistical model is assumed to be a polynomial of the form [44]

$$Y_k\left(\underline{x}\right) \approx a_0 + a_1 x_1 + \cdots + a_n x_n + a_{n+1} x_1^2 + \cdots + a_{2n} x_n^2 + a_{2n+1} x_1 x_2 + \cdots \,, \quad (2.9)$$

where $Y_k\left(\underline{x}\right)$ is the k response, and x_i are the random variables.

As an example, assume the system in Eqn. (2.6). A sensitivity analysis is performed on the system to determine to which of the random variables, α_i, the system is sensitive. Based upon these sensitivities, a simplified model or models is created of Eqn. (2.6). Using a method such as MC analysis or the FOSM or the SOSM, the moments of the simplified models are calculated [13]. Using an assumed statistical model, such as Eqn. (2.9), fit the moments of the RSM models to the assumed model for the outputs.

2.4.2 Assessment

Because an MBD system can be nonlinear, determining the sensitivity of model inputs, or even a simplified model, can be difficult. At best, a local RSM analysis may be performed. In the end, it almost needs to be compared to an MC analysis to ensure that the correct inputs are identified, that the model is simplified correctly, and that a good statistical model is chosen [59]. Due to the choosing of simplified models, it appears as if this is done ad hoc. Different users can apply the method and get different answers. Since RSM is more of a local UA due to the use of a simplified model and violates at least

criterium one in Chapter 1, RSM is not an appropriate method to use with a general MBD system.

2.5 Polynomial Chaos Theory Method

Finally, Polynomial Chaos Theory (PCT) is the last uncertainty analysis. This method is discussed in the following section. PCT then is evaluated for use in an MBD system.

2.5.1 General Approach

The underlying philosophy for PCT, or in some literature generalized Polynomial Chaos (gPC) [27, 43, 61, 63], is that it allows random variables to be represented by a series expansion in terms of orthogonal polynomial basis functions [62]. According to Xiu [61], PCT "is an spectral representation in random space." The PCT expansion was introduced first by Wiener [60] using Gaussian or normally distributed random variables represented by a Hermite orthogonal polynomial basis set. It has been expanded to the Wiener-Askey polynomial chaos, which used other polynomials of the Askey scheme. An Askey scheme is used to classify hypergeometric orthogonal polynomials, of which Hermite polynomials are a subset [62].

A PCT expansion begins with expanding a generic variable based on an

orthogonal polynomial basis set. A second order random process with finite variance can be written as [62]

$$\begin{aligned} X(\theta) =& a_0 I_0 \\ &+ \sum_{i_1=1}^{\infty} c_{i_1} I_1\left(\xi_{i_1}(\theta)\right) \\ &+ \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{\infty} c_{i_1 i_2} I_2\left(\xi_{i_1}(\theta), \xi_{i_2}(\theta)\right) \\ &+ \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{\infty}\sum_{i_3=1}^{\infty} c_{i_1 i_2 i_3} I_3\left(\xi_{i_1}(\theta), \xi_{i_2}(\theta), \xi_{i_3}(\theta)\right) \\ &+ \cdots , \end{aligned} \tag{2.10}$$

where c_i are the coefficients of the expansion, $I_i\left(\xi_i(\theta)\right)$ are the polynomials of the selected orthogonal polynomial basis, $\xi_i(\theta)$ are a vector of random variables, and θ is a random event. More compactly, Eqn. (2.10) can be represented by

$$X = \sum_{j=0}^{\infty} a_j \Psi_j\left(\underline{\xi}\right) , \tag{2.11}$$

where *X* is the random process, a_j are the coefficients of the expansion, Ψ_j are the polynomial of the selected bases and $\underline{\xi}$ are random variables [52]. The orthogonal polynomial basis forms a complete orthogonal basis in the Hilbert space of square integrable random variables [48]

$$\langle \Psi_i, \Psi_j \rangle = 0 \quad \text{for } i \neq j , \tag{2.12}$$

where the operator $\langle \cdot, \cdot \rangle$ is the ensemble average. Also, Eqn. (2.12) can be written as [62]

$$\langle \Psi_i, \Psi_j \rangle = \langle \Psi_i^2 \rangle \delta_{ij} \ , \tag{2.13}$$

where δ_{ij} is the Kronecker delta. To calculate the inner product on the Hilbert space for the random variable ξ [62],

$$\langle f(\xi), g(\xi) \rangle = \int f(\xi) \, g(\xi) \, W(\xi) \, d\xi \ , \tag{2.14}$$

where $W(\xi)$ is the weighting function for the chosen basis set in the Askey scheme [56]. For certain orthogonal polynomials, the weighting functions are probability functions of random distributions [62].

Based on the distribution of the random variables, there may be a better choice for the orthogonal polynomials. Kewlani [27] summarized the preferred orthogonal polynomials basis to use based on a the type of random variable, which can be seen in Table 2.1. A common orthogonal polynomial basis must be used to represent all of the uncertain random variables. If a good match between the orthogonal polynomials and the distribution of variables is not achieved, the expansion still can be performed, but it may require more terms in the series expansion to get the same level of accuracy as if the good match is used [52].

For practical purposes, an infinite expansion is not possible. Given n_v random

Table 2.1: Best orthogonal polynomial basis [27]

Random Variable	Orthogonal Polynomial Basis
Gaussian	Hermite
Gamma	Laguerre
Beta	Jacobi
Uniform	Legendre

variables and n_p, the order of the selected polynomial basis, an appropriate number of terms of the polynomial chaos expansion is [56]

$$P = \frac{(n_p + n_v)!}{n_p! n_v!} - 1. \tag{2.15}$$

This changes the series expansion in Eqn. (2.11) to

$$X \approx \sum_{j=0}^{P} a_j \Psi_j (\xi) \ . \tag{2.16}$$

2.5.2 Polynomial Chaos Theory Process

Each of the random variables is expanded in terms of the selected orthogonal basis. A generic variable a is expanded onto the chosen orthogonal polynomial basis set by [56],

$$a \approx \sum_{i=0}^{n_p} a_i \Psi i \ . \tag{2.17}$$

Each of the random variables needs to be projected onto the orthogonal basis set by using Eqn. (2.14) to determine the coefficients,

$$a_i = \frac{\langle a, \Psi_i \rangle}{\langle \Psi_i \Psi_i \rangle} \, . \tag{2.18}$$

Now with each of the random variables for the system expanded in terms of the orthogonal polynomial basis set, the states of the model also need to be expanded as a function of the orthogonal polynomial basis by Eqn. (2.16). The system of equations contains the same number of equations as it did previously, but are a function of polynomial basis variables, ξ_i. These polynomial basis variables, ξ_i, need to be eliminated to solve the problem. To eliminate them, a multivariate Galerkin projection, a generalization of Eqn. (2.14), is used [62]

$$\langle f\left(\underline{\xi}\right), g\left(\underline{\xi}\right) \rangle = \int f\left(\underline{\xi}\right) g\left(\underline{\xi}\right) W\left(\underline{\xi}\right) d\underline{\xi} \, , \tag{2.19}$$

where $W\left(\underline{\xi}\right)$ is the multivariate weighting function for the chosen orthogonal basis set. The size of the original MBD system is increased and is projected onto the orthogonal polynomial basis.

For a DAE, the resulting system is increased in equations by [48]

$$(P+1)\,(\text{Number of 1st order ODEs}) + (P+1)\,(\text{Number of Algebraic Equations}) \quad (2.20)$$

For the MBD system, the ODEs are second order, creating a state space model of the system results in a system that is

$$2\,(P+1)\,(\text{Number of 2nd order ODEs}) + (P+1)\,(\text{Number of Algebraic Equations}) \quad (2.21)$$

For a simple planar pendulum pinned at one end with the length as a random variable, the original, deterministic system is a DAE with three 2nd order ODEs and two constraint equations. After applying the PCT method, the system consists of six second order, or twelve first order ODEs, and four constraint equations.

2.5.3 Assessment

The power of PCT is that the same information about the outputs can be found from an MC analysis, but in one simulation. This simulation is a larger size, but it only needs to be run once [57]. As an additional benefit, the sensitivity of a given random variable to the output is shown more intuitively. The impact of each of the random

variables is separated into distinct states of the system. In an MC analysis, only the overall impact of all the variables is calculated, not the individual components. For an MC simulation, the system in essence becomes a black box, and post-MC analysis needs to be done to determine the sensitivities. Also, if a three-dimensional rigid or flexible body dynamic system takes hours to run once, then an MC analysis can take days or weeks to run a sufficiently large enough set to get a statistically significant result. With PCT, all the work is up front with computing all of the inner products. Once these are known, the analysis takes significantly less time [57].

PCT offers many benefits with great possibilities. However, PCT also has some drawbacks. Some of the drawbacks involve the complexity of solving for the PCT coefficients. Also, there are some unresolved issues with how it can be applied to a highly nonlinear mechanical system. One issue is that all of the random variables need to be represented by the same polynomial basis [52]. Different polynomial bases are able to represent different types of distributions with fewer terms, which cuts down on the number of inner products that need to be computed. If most of the random variables are of the same type of distribution, then the complexity of solving for PCT coefficient might not be an issue. However, if there are many random variables with many different distributions, then the time savings of a PCT analysis might disappear. An MC analysis might be run in the time it takes to compute all of the PCT coefficients. It is out of the scope of this dissertation to determine the point where PCT becomes too time consuming

when compared to an MC simulation. This point is highly dependent on computer hardware, which is outside of the control of this dissertation.

Another issue with PCT is dealing with nonlinearities. Fisher et al. described two types: polynomial and transcendental nonlinearities [14]. If the polynomial nonlinearities are positive integer order, then the Galerkin projection in Eqn. (2.19) can handle them [14]. If nonlinearity is a rational function of polynomials then through some linear algebra manipulation with the Galerkin projection in Eqn. (2.19), the coefficients can be found. For other special functions, such as the square root, there are special iterative algorithms that can be used to solve for the coefficients [10]. For transcendental nonlinearities, the most popular way to evaluate them is with a Taylor Series expansion [10]. The most common transcendental functions found in an MBD system are trigonometric functions, and the Taylor series expansion is demonstrated in [57]. In the end, there is some fidelity lost because of the linear approximation of the nonlinearity. If the simulation goes too far away from the operating point, the results can be erroneous. It should be noted that both of these uncertainty types are smooth. Another type of uncertainty not really discussed is the discontinuous or piecewise nonlinearity, which can be seen in a mechanical system in the form of friction or contact.

As has been stated before, computing the coefficients needed for PCT is not trivial. The projection of the random variable onto the polynomial basis and the subsequent Galerkin projection to eliminate the basis vector are complicated. The more random

variables that are included in the model, the more complicated these calculations become. In [57] the complicated single and double integral calculations for a two-link robot arm with two uncertainties in each of their lengths are performed.

PCT has been applied specifically to MBD systems in [48, 49, 56, 57]. However, the MBD examples illustrated were all able to be reduced to a set of ODEs. The PCT method was expanded to a maximal set DAE by Ryan [45] for an index-1 DAE MBD system. The index-1 formulation was used because the DAE integrator chosen was Matlab's *ode15i*, which only supports index-1 DAEs. In general, one of the issues with PCT is that all of the work is determining the coefficients by the inner product calculation.

Most of the research on PCT applies it to the state equations; this how PCT is applied to the maximal set DAE by Ryan [45]. Applying it to the state equations means that the deterministic problem is fully derived before the PCT process is initiated. Depending on the system, this can lead to many complicated Galerkin projections. For an MBD system, some of the complexity of the Galerkin projections can be reduced or eliminated. Unfortunately, there is nothing that can be done to reduce the number of inner products required to project the random variables onto the orthogonal basis set. Using a Constrained Lagrangian approach

$$\begin{aligned} &\tfrac{d}{dt}\left(\tfrac{\partial T}{\partial \dot{q}_i}\right) - \tfrac{\partial T}{\partial q_i} + \tfrac{\partial V}{\partial q_i} = Q_i + \mathbf{\Phi}_{q_i}^T \underline{\lambda} \\ &\underline{\Phi} = \underline{0}\ , \end{aligned} \tag{2.22}$$

where T is the kinetic energy, V is the potential energy, Q_i is the ith generalized force, $\mathbf{\Phi}_{q_i}$, is the ith row of the constraint Jacobian, and $\underline{\lambda}$ is the vector of Lagrange multipliers with the constraint equations, $\underline{\Phi} = 0$, some calculations can be reduced. The Galerkin projection of the pieces, such as T and Q, can be performed.

If a Constrained Lagrangian method is applied to already expanded PCT terms, where the orthogonal polynomial basis variables, ξ_i, are eliminated, then a general automated process can be achieved. Since a Constrained Lagrangian process is being used, a robust consistent method for deriving the equations of motion is possible. The process is faster than an MC analysis run and leads directly to uncertainty quantification of the responses. Also, because the responses are a function of each random variable, a sensitivity analysis can be done. A Constrained Lagrangian method lends itself to automation easily. An MBD method can easily be set up using the Constrained Lagrangian. Using PCT to compute the components needed in the Constrained Lagrangian method leads to less complicated PCT coefficient calculations via a Galerkin projection. Constrained Lagrangian derivations need to be placed into state space, which allows a subset of outputs to be calculated for the model and post-processed. All the criteria from Chapter 1 are met. Hence, PCT is be pursued in this dissertation.

In the following chapter, the MBD problem using the Constrained Lagrangian method will be set up and solved. In Chapter 4, PCT will be embedded into the Constrained Lagrangian method.

CHAPTER 3

Set Up and Simulation of a Multi-Body Dynamic System

To automate the derivation and numerical solution of a multi-body dynamic (MBD) system, a set of rigid rules needs to be applied. These rules include the choice of generalized coordinates, the constraint equations, and the form of the final state space model. If these rules and processes are not followed, then an ad hoc method is being built, which is not allowed. The same set of equations needs to be built and solved regardless of the person deriving them.

This chapter derives the theory for an MBD system, including the general structure. Next, the generalized coordinates used for the derivation are explained. Then the holonomic constraints are derived for both two-dimensional (2D) and three-dimensional (3D) formulations. Following this, the Constrained Lagrangian method is explained. The automation process is described next. Finally, the derived MBD system is transformed to state space, and the numerical simulation is explained. This process is illustrated by using a slider-crank example.

3.1 Background of the Multi-Body Dynamics System

For an MBD system, the largest set of equations occur when each body is represented by a set of ODEs equal to the number of unconstrained degrees of freedom

(DOF) in that space plus the kinematic constraint equations. This model uses a maximal set of generalized coordinates [3] or a maximal set of DAEs. Using this approach ensures that a robust set of constraints can be applied to the system. No ad hoc constraints are used, or no reduction of the number of generalized coordinates is allowed. Using a maximal set of generalized coordinates is the approach that many commercial of-the-shelf (COTS) MBD programs use.

Given a maximal set of generalized coordinates, $\underline{q}$, the general form of a rigid-body MBD system is

$$\begin{aligned} \mathbf{M}\left(\underline{q}\right)\underline{\ddot{q}} &= \underline{Q}\left(\underline{\dot{q}},\underline{q},t\right) - \underline{C}\left(\underline{\dot{q}},\underline{q},t\right) + \boldsymbol{\Phi_q}\left(\underline{q},t\right)^T\underline{\lambda} \\ \underline{\Phi}\left(\underline{q},t\right) &= \underline{0}\,, \end{aligned} \tag{3.1}$$

where t is time, $\mathbf{M}\left(\underline{q}\right)$ is the inertia matrix, $\underline{\Phi}\left(\underline{q},t\right)$ are the constraint equations, $\underline{C}\left(\underline{\dot{q}},\underline{q},t\right)$ are the nonlinear acceleration inertial terms, $\boldsymbol{\Phi_q}\left(\underline{q},t\right)$ is the constraint Jacobian, $\underline{\lambda}$ is a vector of the Lagrange multipliers, and $\underline{Q}\left(\underline{\dot{q}},\underline{q},t\right)$ is a vector of the generalized forces. The generalized forces include the conservative forces of the system, the gravitational forces, and spring forces. For simplicity the "$\left(\underline{q}\right)$", "$\left(\underline{q},t\right)$", and "$\left(\underline{\dot{q}},\underline{q},t\right)$" are omitted. A system of this form is a semi-explicit DAE of index-3. The system in Eqn. (3.1) is an index-3 DAE because the constraint equations, $\underline{\Phi}=\underline{0}$, need to be differentiated with respect to time three times to get to a system of ODEs. The system

is semi-explicit because it is a function of the form [15]

$$\begin{aligned} \underline{\ddot{q}} &= \underline{f}\left(\underline{q}, \underline{\dot{q}}, \underline{\lambda}, t\right) \\ \underline{0} &= \underline{g}\left(\underline{q}, \underline{\dot{q}}, t\right). \end{aligned} \tag{3.2}$$

An MBD system of the form in Eqn. (3.1) in general cannot be solved analytically; there is rarely a closed-form solution. Furthermore, numerically solving the system is not easy. First, the systems are very large, nonlinear and stiff. The kinematic constraints guarantee a stiff system because of the infinite frequencies associated with the constraints [4]. A stiff system occurs when the first eigenvalue of the system is much lower (i.e., orders of magnitude lower) than the highest eigenvalue. Second, if using the maximal set of equations for the MBD system, the system needing to be numerically integrated or simulated is a stiff semi-explicit index-3 DAE. To solve this class of problems, a specific type of integrator is needed. Unfortunately, there are not many integrators that solve a stiff semi-explicit index-3 DAE.

3.2 Generalized Coordinates

The generalized coordinates for each body are defined using the maximal set of generalized coordinates for the given space. Approaching the problem with a maximal set allows for a consistent set of robust holonomic constraints to be used. Using a consistent set ensures a non-ad hoc system is built.

This dissertation considers three Euclidean spaces: two-dimensional with no rotation, two-dimensional with rotation SE(2), or full 3D SE(3). The three Euclidean spaces are discussed in this order because it is easier to fully define the problem and take away DOF then it is to add.

3.2.1 Three-Dimensional

For a body defined in a 3D space, the maximal set of generalized coordinates is not straightforward. The translational DOF are Cartesian coordinates and are not an issue in the maximal set. However, there are many options to pick for representing the rotations: Euler Angles, quaternions, Rodriguez Parameters, etc. The advantages and disadvantages of each are outside the scope of this dissertation. However, due to limitations in the Polynomial Chaos Theory (PCT) method, quaternions are used because they result in quadratic terms and not trigonometric (see Section 2.5.3). A quaternion is [22]

$$\underline{e} = \begin{bmatrix} e0 \\ e1 \\ e2 \\ e3 \end{bmatrix} . \tag{3.3}$$

Given as a screw axis, an instantaneous axis of rotation, $\underline{\omega}$, and a rotation about that axis, θ, the physical meaning behind a quaternion can be thought of as [39]

$$\underline{e} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ \underline{\omega}\sin\left(\frac{\theta}{2}\right) \end{bmatrix}. \tag{3.4}$$

This relationship is only true if the quaternions being used are unit quaternions [39]. To be a unit quaternion, the components must satisfy

$$\Phi_e = e0^2 + e1^2 + e2^2 + e3^2 = 1. \tag{3.5}$$

Thus, for a body defined in a 3D space using quaternions, the maximal set of generalized coordinates for body *i* is

$$\underline{q}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \\ e0_i \\ e1_i \\ e2_i \\ e3_i \end{bmatrix}. \tag{3.6}$$

Using this set of generalized coordinates means that a constraint equation enforcing that the quaternions are unit quaternions is needed, Eqn.(3.5).

Useful Quaternion Identities

To use quaternions as generalized coordinates, two useful identities are needed. The first identity, $\mathbf{A}_i$, defines general constraints and coordinate transformations. Given $\underline{e}_i$, a rotation matrix is [22]

$$\mathbf{A}_i = \begin{bmatrix} 2\,e0_i{}^2 + 2\,e1_i{}^2 - 1 & -2\,e0_i\,e3_i + 2\,e1_i\,e2_i & 2\,e0_i\,e2_i + 2\,e1_i\,e3_i \\ 2\,e0_i\,e3_i + 2\,e1_i\,e2_i & 2\,e0_i{}^2 + 2\,e2_i{}^2 - 1 & -2\,e0_i\,e1_i + 2\,e2_i\,e3_i \\ -2\,e0_i\,e2_i + 2\,e1_i\,e3_i & 2\,e0_i\,e1_i + 2\,e2_i\,e3_i & 2\,e0_i{}^2 + 2\,e3_i{}^2 - 1 \end{bmatrix}. \tag{3.7}$$

To relate body-fixed angular velocity to quaternions [22]

$$\underline{\omega}_i = 2\mathbf{G}_i \begin{bmatrix} \dot{e0}_i \\ \dot{e1}_i \\ \dot{e2}_i \\ \dot{e3}_i \end{bmatrix}, \tag{3.8}$$

where

$$\mathbf{G}_i = \begin{bmatrix} -e1_i & e0_i & e3_i & -e2_i \\ -e2_i & -e3_i & e0_i & e1_i \\ -e3_i & e2_i & -e1_i & e0_i \end{bmatrix} . \tag{3.9}$$

Equation (3.8) is useful in defining the kinetic energy for the rotational DOFs with quaternions.

3.2.2 Two-Dimensional with Rotation

For a body defined in a 2D space with rotation, the maximal set of generalized coordinates for body *i* is typically

$$\underline{q}_i = \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix} . \tag{3.10}$$

θ_i is defined as positive about the *z*-axis of the system, perpendicular to the plane of translation. θ_i ultimately results in trigonometric functions in the equations of motion, which due to a limitation in PCT, needs to be approximated by a Taylor series [10] (see Section 2.5.3). This limitation can be overcome by using quaternions or Euler parameters [22] which result in quadratic terms, not trigonometric. Because this is a 2D problem with rotation only about the *z*-axis, the quaternions in Eqn. (3.3) simplify to

$$\underline{e} = \begin{bmatrix} e0 \\ \\ e3 \end{bmatrix} . \tag{3.11}$$

The resulting unit quaternion constraint for 2D is

$$\Phi_e = e0^2 + e3^2 = 1. \tag{3.12}$$

Substituting the unit quaternions for θ_i, the resulting maximal set of generalized coordinates for body *i* is

$$\underline{q}_i = \begin{bmatrix} x_i \\ \\ y_i \\ \\ e0_i \\ \\ e3_i \end{bmatrix} . \tag{3.13}$$

Useful Quaternion Identities

To use quaternions as generalized coordinates in 2D, two useful identities are needed. The first identity, $\mathbf{A}_i$, helps define general constraints and coordinate transformations. Given $\underline{e}_i$, a rotation matrix is

$$\mathbf{A}_i = \begin{bmatrix} 2\,e0_i{}^2 - 1 & -2\,e0_i\,e3_i \\ \\ 2\,e0_i\,e3_i & 2\,e0_i{}^2 - 1 \end{bmatrix} . \tag{3.14}$$

To relate body-fixed angular velocity to quaternions,

$$\underline{\omega}_i = 2\mathbf{G}_i \begin{bmatrix} \dot{e0}_i \\ \\ \dot{e3}_i \end{bmatrix} , \tag{3.15}$$

where

$$\mathbf{G}_i = \begin{bmatrix} -e3_i & e0_i \end{bmatrix} . \tag{3.16}$$

Equation (3.15) is useful in defining the kinetic energy for the rotational DOFs.

3.2.3 Two-Dimensional with No Rotation

For a body defined in a 2D space with no rotation, the maximal set of generalized coordinates for body *i* is

$$\underline{q}_i = \begin{bmatrix} x_i \\ \\ y_i \end{bmatrix} . \tag{3.17}$$

There are no rotational DOFs in this case.

3.3 Holonomic Constraints

With the description of the generalized coordinates of each body defined, a formulation of how these bodies interact with each other via kinematic constraints is needed. To ensure that the method used to build the MBD system is not ad hoc, a set of robust holonomic constraints needs to be defined. These constraints help with the

automation by applying consistency to all of the kinematics constraints. To build the holonomic constraint equations for the system, there are two categories: constraints and joints. Constraints are simple kinematic position constraints that typically only constrain one or two DOF. Joints typically are made up of multiple position kinematic constraints and form complex kinematic constraints.

For the purpose of this dissertation, the axis used to define constraints and joints is a z-axis of a coordinate system located on the respective bodies for 3D and a y-axis for 2D. Also body *i* is constrained to follow body *j*, $i \neq j$. The vectors and rotation matrices for a given body used for the definition of the constraints can be seen in Fig. 3.1. The rotation matrix for the origin of body *i* with respect to the inertial or global frame is denoted $\mathbf{A}_i$. The rotation matrix from the body frame of *i* to the frame on body *i* that forms the constraint is $\mathbf{A}'_i$. $\mathbf{A}_i$ is a function of the generalized coordinates, while $\mathbf{A}'_i$ is constant. The point from the origin to the origin of body *i* is denoted $\underline{r}_i$, and the point from the origin of body *i* to the constraint application point is $\underline{s}'_i$.

A summary of the constraints and joints is presented first, followed by a more formal definition. The constraints and joints follow Haug's definitions [22]. Some of the constraints and joints work for both 2D and 3D Euclidean spaces, others are only for 3D. The constraints are:

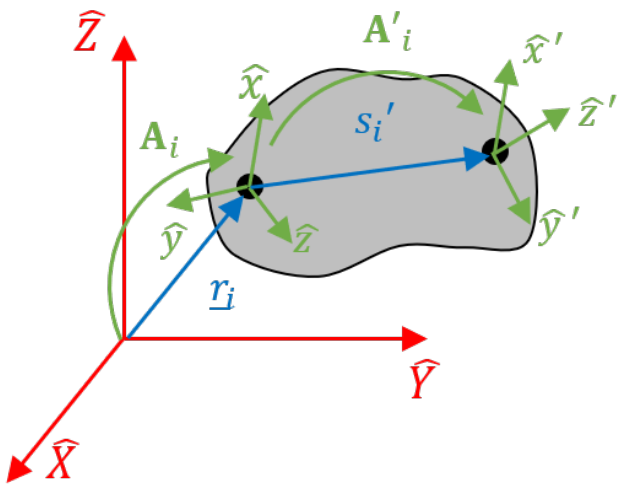

Figure 3.1: Part vector and rotation matrix description

- Dot-1 – an axis on one body remains perpendicular to an axis on another body (2D and 3D)
- Dot-2 - translation along an axis remains perpendicular to axis on another body (2D and 3D)
- Parallel-1 – an axis on one body remains parallel to an axis on another body (3D only)
- Parallel-2 – the distance between a point on each body remains parallel to an axis on one body (3D only)
- Fixed Point – a point on one body is fixed to another body (2D and 3D)
- Fixed Distance – distance between a point on each body remains fixed (2D and 3D)

The joints, which are composites of multiple constraints, are:

- Fixed – no net motion between two bodies (2D and 3D)
- Revolute – one relative rotational DOF between two bodies (2D and 3D)
- Prismatic – one relative translational DOF between two bodies (2D and 3D)
- Spherical – translational DOF fixed between two bodies (3D only)
- Cylindrical - one relative rotational DOF between two bodies and one translational DOF about axis (3D only)
- Universal – two rotational DOF between two bodies (3D only)

3.3.1 Three-Dimensional Constraints

A system of constraints is needed to use in the 3D formulation. The constraints outlined follow Haug's definitions [22].

- The first constraint needed is a dot-1 constraint, where one z-axis in one coordinate system on body *i* stays perpendicular to a z-axis in the other coordinate system on body *j* [22],

$$\Phi_{d1}\left(\mathbf{A}_i, \mathbf{A}'_i, \mathbf{A}_j, \mathbf{A}'_j\right) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \left(\mathbf{A}_i \mathbf{A}'_i\right)^T \left(\mathbf{A}_j \mathbf{A}'_j\right) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} . \tag{3.18}$$

- The next constraint is the fixed point constraint. This constraint fixes a point on

body *i* to a point on body *j*. It is defined as [22]

$$\Phi_{fp}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j\right) = \left(\underline{r}_i + \mathbf{A}_i\underline{s}'_i\right) - \left(\underline{r}_j + \mathbf{A}_j\underline{s}'_j\right) \ . \tag{3.19}$$

- The next constraint is a dot-2 constraint. This constraint ensures that the translation along an axis remains perpendicular to an axis on body *j*, [22]

$$\Phi_{d2}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j, \mathbf{A}'_j\right) = \left(\mathbf{A}_j\mathbf{A}'_j\right)^T \Phi_{fp}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j\right) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \ . \tag{3.20}$$

- The next constraint is a parallel-1 constraint. This constraint ensures that the *z*-axis on body *i* remains parallel to a *z*-axis on body *j*, [22]

$$\Phi_{p1}\left(\mathbf{A}_i, \mathbf{A}'_i, \mathbf{A}_j, \mathbf{A}'_j\right) = \begin{bmatrix} \Phi_{d1}\left(\mathbf{A}_i, \mathbf{A}'_i R\left(x, \frac{-\pi}{2}\right), \mathbf{A}_j, \mathbf{A}'_j\right) \\ \Phi_{d1}\left(\mathbf{A}_i, \mathbf{A}'_i R\left(y, \frac{\pi}{2}\right), \mathbf{A}_j, \mathbf{A}'_j\right) \end{bmatrix} , \tag{3.21}$$

where $R\left(\text{axis}, \text{angle}\right)$ denotes a rotation about an axis by an angle.

- The next constraint is a parallel-2 constraint. This constraint ensures that the distance between a point on each of the bodies remains parallel to a *z*-axis on body

j, [22]

$$\Phi_{p2}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j, \mathbf{A}'_j\right) = \begin{bmatrix} \Phi_{d2}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j, \mathbf{A}'_j R\left(x, \frac{-\pi}{2}\right)\right) \\ \Phi_{d2}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j, \mathbf{A}'_j R\left(y, \frac{\pi}{2}\right)\right) \end{bmatrix} . \quad (3.22)$$

- Finally, the last constraint is a fixed distance constraint that typically is formed by two spherical joints [22]

$$\Phi_{fd}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j, d\right) = \Phi_{fp}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j\right)^T \Phi_{fp}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j\right) - d^2 , \quad (3.23)$$

where d is the fixed distance between the two spherical joints.

These six constraints are the building blocks for joints. Using these constraints in different combinations results in standard joints such as a pin joint or a spherical joint.

3.3.2 Three-Dimensional Joints

In a mechanical system, it is rare that parts are constrained by one or two constraints. To help in the definition of how parts or bodies in a mechanical system are more commonly constrained, joints are used.

- A fixed joint is a joint which allows zero DOFs between two parts. The constraints defining this joint are

$$\Phi_{\text{fix}}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \mathbf{A}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j, \mathbf{A}'_j\right) = \begin{bmatrix} \Phi_{fp}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j\right) \\ \Phi_{p1}\left(\mathbf{A}_i, \mathbf{A}'_i, \mathbf{A}_j, \mathbf{A}'_j\right) \\ \Phi_{d1}\left(\mathbf{A}_i, \mathbf{A}'_i R\left(y, \frac{\pi}{2}\right), \mathbf{A}_j, \mathbf{A}'_j R\left(x, \frac{-\pi}{2}\right)\right) \end{bmatrix} . \quad (3.24)$$

- A revolute joint is a joint which allows one relative rotational DOF between two parts. This joint is sometimes known as a pin or hinge joint. The constraints defining this joint are [22]

$$\Phi_{\text{rev}}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \mathbf{A}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j, \mathbf{A}'_j\right) = \begin{bmatrix} \Phi_{fp}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j\right) \\ \Phi_{p1}\left(\mathbf{A}_i, \mathbf{A}'_i, \mathbf{A}_j, \mathbf{A}'_j\right) \end{bmatrix} . \quad (3.25)$$

- A spherical joint is a joint which allows all translations between two parts at a point to be fixed, but constrains zero rotational DOFs. The constraints defining this joint are [22]

$$\Phi_{\text{sph}}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j\right) = \Phi_{fp}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j\right) . \quad (3.26)$$

- A prismatic joint, or sometimes a translational joint, is a joint which allows one relative translational DOF between two parts. The constraints defining this joint

are [22]

$$\Phi_{\text{pris}}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \mathbf{A}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j, \mathbf{A}'_j\right) = \begin{bmatrix} \Phi_{p1}\left(\mathbf{A}_i, \mathbf{A}'_i, \mathbf{A}_j, \mathbf{A}'_j\right) \\ \Phi_{p2}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j, \mathbf{A}'_j\right) \\ \Phi_{d1}\left(\mathbf{A}_i, \mathbf{A}'_i R\left(y, \frac{\pi}{2}\right), \mathbf{A}_j, \mathbf{A}'_j R\left(x, \frac{-\pi}{2}\right)\right) \end{bmatrix}. \quad (3.27)$$

- A cylindrical joint is a joint which allows one relative rotational DOF between two parts and one translational DOF between those two parts along the rotational axis. The constraints defining this joint are [22]

$$\Phi_{\text{cyl}}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \mathbf{A}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j, \mathbf{A}'_j\right) = \begin{bmatrix} \Phi_{p1}\left(\mathbf{A}_i, \mathbf{A}'_i, \mathbf{A}_j, \mathbf{A}'_j\right) \\ \Phi_{p2}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j, \mathbf{A}'_j\right) \end{bmatrix}. \quad (3.28)$$

- A universal or Hooke's joint is a joint which allows two rotational DOFs between two parts, and fixes all translational DOFs. The constraints defining this joint are [22]

$$\Phi_{\text{univ}}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \mathbf{A}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j, \mathbf{A}'_j\right) = \begin{bmatrix} \Phi_{fp}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j\right) \\ \Phi_{d1}\left(\mathbf{A}_i, \mathbf{A}'_i, \mathbf{A}_j, \mathbf{A}'_j\right) \end{bmatrix}. \quad (3.29)$$

These joints are not all the types of joints that can be defined, but include all the joints

used in this dissertation. The constraints and joints described in 3D are the typical types that are found in COTS software. With 3D constraints and joints defined, a subset of these is used to define constraints and joints for a 2D formulation.

3.3.3 Two-Dimensional Constraints

A system of constraints is needed to use in the 2D formulation, similar to the 3D constraints. The 2D constraints outlined follow Haug's definitions [22]. These differ in the 3D definitions by using the *y*-axis instead of the *z*-axis. The *y*-axis is used in 2D because the *z*-axis is technically always perpendicular to the plane and is not available as a DOF to constrain. Note, the dimensionality of the vectors in 2D are 2 by 1 and matrices are 2 by 2.

- The first constraint needed is a dot-1 constraint, where one *y*-axis in one coordinate system on body *i* stays perpendicular to a *y*-axis in the other coordinate system on body *j* [22],

$$\Phi_{d1}\left(\mathbf{A}_i, \mathbf{A}'_i, \mathbf{A}_j, \mathbf{A}'_j\right) = \begin{bmatrix} 0 & 1 \end{bmatrix} \left(\mathbf{A}_i \mathbf{A}'_i\right)^T \left(\mathbf{A}_j \mathbf{A}'_j\right) \begin{bmatrix} 0 \\ 1 \end{bmatrix} . \tag{3.30}$$

- The fixed point constraint in 2D is the same as 3D and is defined in Eqn. (3.19).

- The dot-2 constraint for 2D is defined as [22]

$$\Phi_{d2}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j, \mathbf{A}'_j\right) = \left(\mathbf{A}_j \mathbf{A}'_j\right)^T \Phi_{fp}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j\right) \begin{bmatrix} 0 \\ \\ 1 \end{bmatrix} . \quad (3.31)$$

- The fixed distance constraint in 2D is defined between two revolute joints and is defined using Eqn. (3.23).

Similar to the 3D formulation, these four constraints are the building blocks for joints in 2D.

3.3.4 Two-Dimensional Joints

For a planar system, the joint definitions of how parts or bodies in a mechanical system are less than in 3D.

- The fixed joint has the same definition as 3D, but constraints are slightly different. The constraints defining this joint are

$$\Phi_{\text{fix}}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j\right) = \begin{bmatrix} \Phi_{fp}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j\right) \\ \\ \Phi_{d1}\left(\mathbf{A}_i, \mathbf{A}'_i R\left(y, \frac{\pi}{2}\right), \mathbf{A}_j, \mathbf{A}'_j R\left(x, \frac{-\pi}{2}\right)\right) \end{bmatrix} . \quad (3.32)$$

- The revolute joint has the same definition as 3D, but constraints are slightly

different. The constraints defining this joint are [22]

$$\Phi_{\text{rev}}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j\right) = \Phi_{fp}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j\right). \tag{3.33}$$

- The prismatic joint has the same definition as 3D, but constraints are slightly different. The constraints defining this joint are [22]

$$\Phi_{\text{pris}}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \mathbf{A}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j, \mathbf{A}'_j\right) = \Phi_{d2}\left(\underline{r}_i, \mathbf{A}_i, \underline{s}'_i, \underline{r}_j, \mathbf{A}_j, \underline{s}'_j, \mathbf{A}'_j\right). \tag{3.34}$$

Having constraints and joints defined for both 3D and 2D formulations of an MBD system, attention can be shifted to the derivation of the equations of motion.

3.4 Equations of Motion Derivation

With the generalized coordinates and the systematic process for constraint equations chosen, the equations of motion can be derived. The equations of motion of an MBD system are most easily automated using a Constrained Lagrangian approach. Other methods can be used such as Kane's Method, Gibbs-Appell, or Newton-Euler [17, 18], but these are not as straightforward or as easy to automate as a Lagrangian approach. The

Constrained Lagrangian equations of motion are derived using

$$\begin{aligned} &\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) - \frac{\partial T}{\partial q_i} + \frac{\partial V}{\partial q_i} = Q_i + \mathbf{\Phi}_{q_i}^T \underline{\lambda} \\ &\underline{\Phi} = \underline{0}\,, \end{aligned} \tag{3.35}$$

where q_i are generalized coordinates, T is the kinetic energy of the system, V is the potential energy of the system, Q_i are the generalized forces, $\underline{\lambda}$ are a vector of the Lagrange multipliers, $\mathbf{\Phi}_{q_i}$, is the *ith* row of the constraint Jacobian, and $\underline{\Phi}$ are the constraint equations. Using Eqn. (3.35) leads to Eqn. (3.1). Building the components of Eqn. (3.35 are easy to automate with a symbolic mathematical software such as Maple.[1]

To solve the MBD system in Eqn. (3.1), the various vectors and matrices need to be found using the Constrained Lagrangian method. With the components found, the system can be solved or simulated numerically.

[1]Note that the Constrained Lagrangian approach is more constraining in its choice of states for a system. In other methods, such as Kane's method, are position states are the generalized coordinates, $\underline{q}$, and the velocity states are generalized speeds, $\underline{u}$ [26]. To use the Constrained Lagrangian method, $u_i = \dot{q}_i$. Other methods do not carry this restriction.

3.5 Numerical Solution

To solve the problem, Eqn. (3.1) needs to be transformed into state space. The states of a system derived using the Constrained Lagrangian method are

$$\underline{x} = \begin{bmatrix} \underline{q} \\ \underline{\dot{q}} \\ \underline{\lambda} \end{bmatrix} = \begin{bmatrix} \underline{x}_p \\ \underline{x}_v \\ \underline{x}_\lambda \end{bmatrix} . \tag{3.36}$$

The use of the variable "x" is customary when placing a system into state space. The variable is different from the generalized coordinate DOF, "x_i". Substituting Eqn. (3.36) into Eqn. (3.1) yields the following state space model for an MBD system

$$\begin{aligned} \underline{\dot{x}}_p &= \underline{x}_v \\ \mathbf{M}\underline{\dot{x}}_v &= \underline{Q} - \underline{C} + \mathbf{\Phi}_{\boldsymbol{q}}^T \underline{x}_\lambda \\ \underline{\Phi} &= \underline{0} \, . \end{aligned} \tag{3.37}$$

This system is an index-3 DAE because there are position constraints. The constraints need to be differentiated three times to get to a system of ODEs.

There are not many index-3 DAE solvers available to solve the system in Eqn. (3.37). To help find solvers for this problem, the index of the system needs to be reduced. In theory, differentiating $\underline{\Phi}$ with respect to time once or twice leads to an index-2 or

index-1 DAE, respectively. However, these systems can have numerical issues because there are no constraints enforcing the position constraints; over time the system can drift.

A better way to help with the solver issue is to keep the position constraints in the system. One can keep the position constraints explicitly defined and reduce the index by introducing additional Lagrange multipliers, $\underline{\mu}$ [16]. This approach ensures that the position constraints are enforced [55]. The system states then become

$$\underline{x} = \begin{bmatrix} \underline{x}_p \\ \underline{x}_v \\ \underline{x}_\lambda \\ \underline{x}_\mu \end{bmatrix} , \tag{3.38}$$

where $\underline{x}_\mu = \underline{\mu}$. These added states change Eqn. (3.37) to

$$\begin{aligned} \underline{\dot{x}}_p &= \underline{x}_v + \mathbf{\Phi}_{\boldsymbol{q}}^T \underline{x}_\mu \\ \mathbf{M}\underline{\dot{x}}_v &= \underline{Q} - \underline{C} + \mathbf{\Phi}_{\boldsymbol{q}}^T \underline{x}_\lambda \\ \underline{\Phi} &= \underline{0} \\ \mathbf{\Phi}_{\boldsymbol{q}} \underline{x}_v &= -\underline{\Phi}_t \ , \end{aligned} \tag{3.39}$$

where $\underline{\Phi}_t$ is a vector of the position constraints explicitly differentiated with respect to time. This index-2 DAE system is referred to as a stabilized index-2 (SI2) DAE because it ensures that the position constraints do not drift [55]. In essence, the index-3 DAE is

embedded in the index-2 DAE. The system can be reduced further to an index-1 by repeating the same reduction method [16]; however such an index reduction is not necessary.

The system in Eqn. (3.39) is simulated using Matlab using a DAE solver. Matlab is chosen because it is easier to debug and simulate systems then more general programming languages such as C or Fortran. Unfortunately, Matlab only supports index-1 DAEs [33]. To solve the system in Eqn. (3.39), the Matlab toolbox, *sundialsTB*, from Lawrence Livermore National Laboratory [23] is used. The integrator *IDAS* from *sundialsTB* supports index-1 or higher DAEs [24].

Before the system is ready to be solved, a consistent set of initial conditions needs to be found. A consistent set of initial conditions ensures that the residual at $t = 0$ of the system in Eqn. (3.39) is zero. Rearranging, the residual equation for Eqn. (3.37) is

$$\begin{aligned} -\underline{\dot{x}}_p + \underline{x}_v + \mathbf{\Phi}_{\boldsymbol{q}}^T \underline{x}_\mu &= \underline{0} \\ -\mathbf{M}\underline{\dot{x}}_v + \underline{Q} - \underline{C} + \mathbf{\Phi}_{\boldsymbol{q}}^T \underline{x}_\lambda &= \underline{0} \\ \underline{\Phi} &= \underline{0} \\ \mathbf{\Phi}_{\boldsymbol{q}} \underline{x}_v + \underline{\Phi}_t &= \underline{0}\,. \end{aligned} \tag{3.40}$$

To find a consistent set of initial conditions, a rudimentary assembly (i.e., root-finding) algorithm is used for the system in Eqn. (3.39). The rudimentary assembly algorithm uses a Newton-Raphson method to find $\underline{x}_{p0}$ and $\underline{x}_{v0}$, where the subscript "0"

denotes initial value. Using the position constraint equations from Eqn. (3.39), the specified position initial conditions are substituted. The Newton-Raphson method requires the computation of the system Jacobian, which has already been computed, $\mathbf{\Phi}_q$. The rows and columns pertaining to the specified initial conditions are removed. The Newton-Raphson method calculates the remaining consistent set of positional initial conditions, $\underline{x}_{p0}$.

To find the consistent set of velocity initial conditions, $\underline{x}_{v0}$, the process is repeated with the velocity constraint equations from Eqn. (3.39). Only the constraint Jacobian, $\mathbf{\Phi}_q$, and $\underline{\Phi}_t$ are needed, and the rows and columns pertaining to the specified initial conditions are removed from the constraint Jacobian for the Newton-Raphson Jacobian.

Finally, the accelerations and Lagrange Multiplier initial states need to be calculated, using the index-1 DAE system. The index-1 DAE uses acceleration constraints instead of positions constraints. The index-1 DAE state space model is

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{M} & -\mathbf{\Phi}_q^T \\ \mathbf{0} & \mathbf{\Phi}_q & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\underline{x}}_p \\ \dot{\underline{x}}_v \\ \dot{\underline{x}}_\lambda \end{bmatrix} = \begin{bmatrix} \underline{x_v} \\ \underline{Q} \\ \underline{\gamma} - \underline{\Phi}_{tt} \end{bmatrix}, \tag{3.41}$$

where $\mathbf{0}$ is a matrix of zeros, $\underline{\gamma}$ is the nonlinear acceleration constraint terms, and $\underline{\Phi}_{tt}$ is the second derivative explicitly with respect to time of the constraints equations. Note, $\dot{\underline{x}}_\mu = \underline{0}$ by definition [16].

The MBD system has been fully derived. The next step is to automate the process using the generalized coordinates, the constraint and joint equations, and the Constrained Lagrangian method.

3.6 Automation

Now with rules and processes for the MBD system set, the automation of the derivation and simulation can be considered. The flow chart of the Deterministic Multi-Body Dynamics (DMBoD) automation process can be seen in Fig. 3.2. The process starts with Maple to calculate symbolically each of the matrices and vectors needed to set up Eqn. (3.39). These matrices and vectors are exported as Matlab functions. In Matlab, the minimum set of initial conditions are specified, and the system is simulated.

3.6.1 Inputs

To set up the dimensionality of the system to be simulated, the following inputs are needed:

- The gravitational vector, $\underline{g}$, is needed.
- The number of generalized coordinates per body n_{dof}. This number is either 2, 4, or 7 generalized coordinates, depending on the dimensionality of the model. Each of

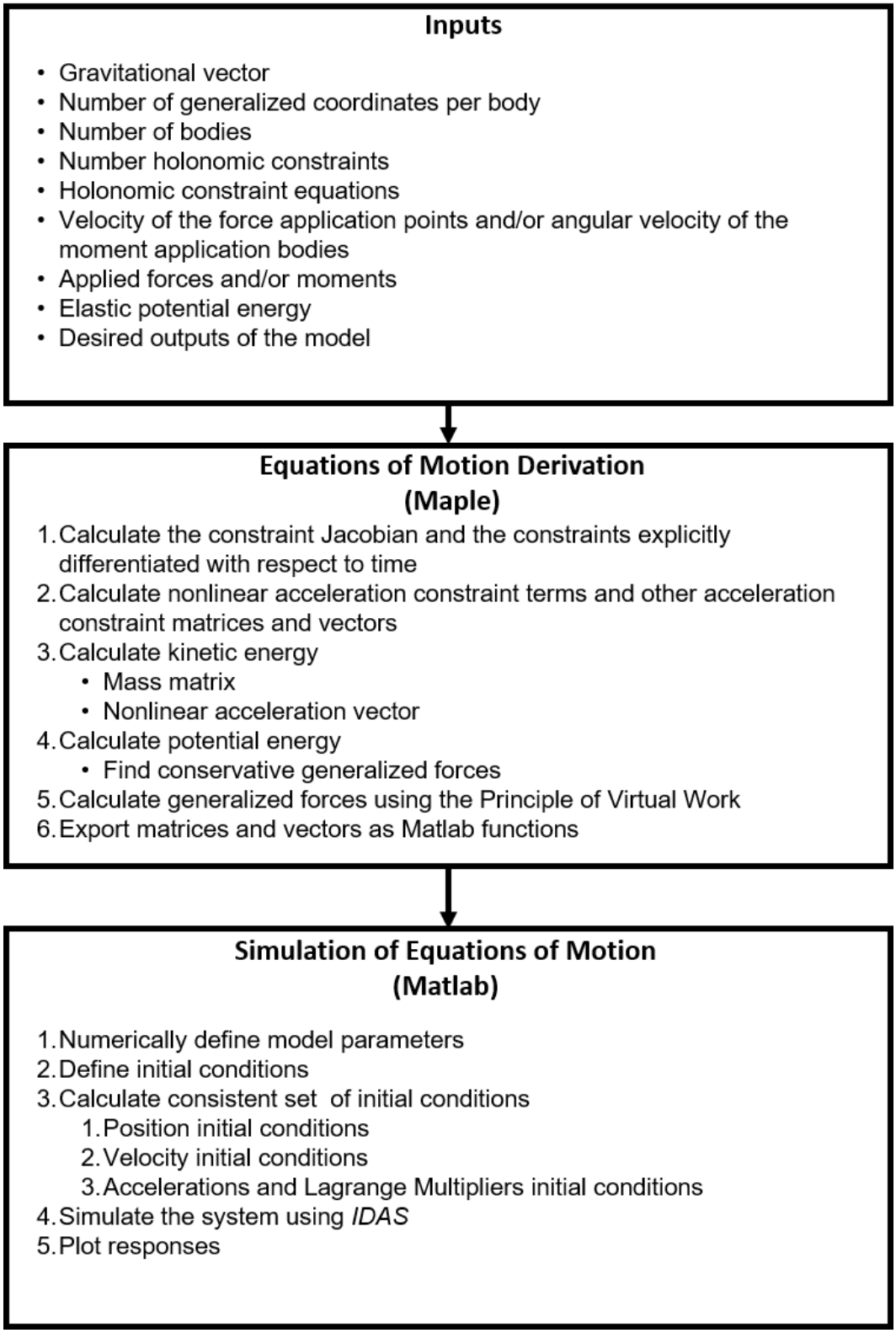

Figure 3.2: DMBoD process flow chart

these cases is defined in Section 3.2, specifically Section 3.2.3, Section 3.2.2, and Section 3.2.1, respectively.

- The number of bodies in the model, n_{bod}, is needed
- The number of holonomic constraints, n_{con}, is needed.

Using the holonomic constraint definitions in Section 3.3, the constraint vector, $\underline{\Phi}$, is defined. Each of the applied forces and moments is defined along with the corresponding velocity variable of the point for a force or the angular velocity variable of the body for a moment. Then the elastic potential energy is defined. As an example, the elastic potential energy for a system with linear springs is

$$V_{\mathrm{elastic}} = \sum_{i=1}^{n_{\mathrm{spr}}} \frac{1}{2} d_i^T \mathbf{k}_i d_i \ , \tag{3.42}$$

where n_{spr} is the number of springs in the system, d_i is the distance the spring is compressed, and $\mathbf{k}_i$ is the spring stiffness matrix. The linear spring's elastic potential is shown as one example, and the elastic potential easily can contain nonlinear stiffness terms. Finally, the desired responses from the model, the outputs, are defined.

3.6.2 Maple

With the dimensionality of the system defined, the kinematic constraints defined, the forces and moments and the velocity and angular velocity of their application defined,

and the elastic potential energy, the components of Eqn. (3.39) can be calculated. The following calculations are all automated using custom-built Maple libraries.

First dealing with the constraint equations, the constraint Jacobian, $\mathbf{\Phi}_q$, and the constraint vector differentiated explicitly with respect to time, $\underline{\Phi}_t$, are calculated. Next, the acceleration components of the constraint equations are found.

The energy of the system is the next step in the automation. The kinetic energy of the system is calculated by

$$T = \sum_{i=1}^{n_{\text{bod}}} \frac{1}{2} \underline{\dot{q}}_i^T \mathbf{M}_i \underline{\dot{q}}_i \ , \tag{3.43}$$

where

$$\mathbf{M}_i = \begin{bmatrix} m_i \mathbf{I} & \mathbf{0} \\ \mathbf{0} & 4\mathbf{G}_i^T \mathbf{J}_i \mathbf{G}_i \end{bmatrix} \ , \tag{3.44}$$

m_i is the mass of body i, $\mathbf{I}$ is the identity matrix, and $\mathbf{J}_i$ is the inertia tensor of body i. When applying the Constrained Lagrangian with n generalized coordinates, this leads to $\mathbf{M}$ and $\underline{C}$ in Eqn. (3.39) by

$$\mathbf{M} = \begin{bmatrix} \frac{\partial}{\partial \ddot{q}_1}\left(\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_1}\right)\right) & \frac{\partial}{\partial \ddot{q}_2}\left(\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_1}\right)\right) & \cdots & \frac{\partial}{\partial \ddot{q}_n}\left(\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_1}\right)\right) \\ \frac{\partial}{\partial \ddot{q}_1}\left(\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_2}\right)\right) & \frac{\partial}{\partial \ddot{q}_2}\left(\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_2}\right)\right) & \cdots & \frac{\partial}{\partial \ddot{q}_n}\left(\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_2}\right)\right) \\ \vdots & \vdots & & \vdots \\ \frac{\partial}{\partial \ddot{q}_1}\left(\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_n}\right)\right) & \frac{\partial}{\partial \ddot{q}_2}\left(\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_n}\right)\right) & \cdots & \frac{\partial}{\partial \ddot{q}_n}\left(\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_n}\right)\right) \end{bmatrix} \ , \tag{3.45}$$

and

$$\underline{C} = \begin{bmatrix} \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_1}\right) - \frac{\partial T}{\partial q_1} \\ \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_2}\right) - \frac{\partial T}{\partial q_2} \\ \vdots \\ \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_n}\right) - \frac{\partial T}{\partial q_n} \end{bmatrix} - \mathbf{M}\underline{\ddot{q}} \ . \tag{3.46}$$

$\mathbf{M}$ is the matrix that pre-multiplies all of the linear acceleration terms and is not made up directly of $\mathbf{M}_i$. The potential energy for the system has two components, the gravitational potential and the elastic potential. The gravitational potential is

$$V_{\text{grav}} = \sum_{i=1}^{n_{\text{bod}}} -m_i \underline{g}^T \cdot \underline{q}_{Ti} \ , \tag{3.47}$$

where $\underline{g}$ is the gravitational vector defined in the inertial frame, and $\underline{q}_{Ti}$ are the translational generalized coordinates for a body. This formulation is slightly different from the typical $V = m_i g h$ definition because gravity is defined as a vector in global coordinates. The total potential energy for the system to be used in Eqn. (3.35) is

$$V = V_{\text{grav}} + V_{\text{elastic}} \ , \tag{3.48}$$

recalling that V_{elastic} is defined by the user. Finally, the generalized forces, Q_i, are found with help from the Principle of Virtual Work,

$$\delta\text{Work} = \sum_{i=1}^{n_{\text{frc}}} \underline{F}_i \cdot \delta\underline{r}_i + \sum_{j=1}^{n_{\text{mom}}} \underline{M}_j \cdot \delta\underline{\theta}_j \ , \tag{3.49}$$

where n_{frc} is the number of forces, n_{mom} is the number of moments, $\underline{F}_i$ is a force, $\underline{M}_j$ is a moment, $\delta\underline{r}_i$ is the virtual displacement of the force, $\delta\underline{\theta}_j$ is the virtual displacement of the moment, and $\delta\mathrm{Work}$ is the virtual work. The virtual displacements, $\delta\underline{r}_i$ and $\delta\underline{\theta}_j$, are computed by using the kinematical method [17]. In the kinematical method, the velocity of a point or the angular velocity of a body is computed as functions of the generalized coordinates and time. Then the generalized coordinate velocities, $\dot{q}_i$, are replaced with the virtual displacement of the generalized coordinates, δq_i.

To find the generalized forces, the potential energy in Eqn. (3.48) and the virtual work in Eqn. (3.49) are used. The generalized force for generalized coordinate *k* is

$$Q_k = \frac{\partial\left(\delta\mathrm{Work}\right)}{\partial\left(\delta q_k\right)} - \frac{\partial V}{\partial q_k}\,. \tag{3.50}$$

Using the Constrained Lagrangian in Eqn. (3.35), the various matrices and vectors are calculated for Eqn. (3.39) and exported as Matlab functions. The system is built with custom Maple functions. The descriptions of the functions can be found in Appendix A.

3.6.3 Matlab

The last step in the automation process in Fig. 3.2 is to solve the system in Matlab by simulating through time. With the components of the equations exported from Maple as Matlab functions, the final input are the initial conditions for the free DOFs of the

system. Matlab then solves for a consistent set of initial conditions following the process given in Section 3.5. Finally, the system in Eqn. (3.39) is simulated, and the outputs are plotted. The system is simulated with custom Matlab functions. The descriptions of the functions can be found in Appendix B.

3.6.4 Automation Summary

The method followed in this dissertation to automate an MBD system is not entirely new. Most COTS programs do not need to derive the equations for each solve because the number of generalized coordinates per body are fixed. The components of Eqn. (3.39) can be solved for up front for any generic problem, and the symbolic derivation from Maple is not needed for every problem. In the PCT implementation in the next chapter, the number of generalized coordinates per body changes depending on the number of random variables and the orthogonal polynomial basis. Each problem is unique due to the number of random variables and the number of generalized coordinates per body. Therefore, the components of Eqn. (3.39) need to be derived for each problem.

3.7 Example: Deterministic Slider-Crank

As an example to show how the DMBoD is applied to a problem, a slider-crank mechanism based on the work from [57] is used. The slider-crank equations of motion are derived, and the various components of Eqn. (3.39) are calculated and exported.

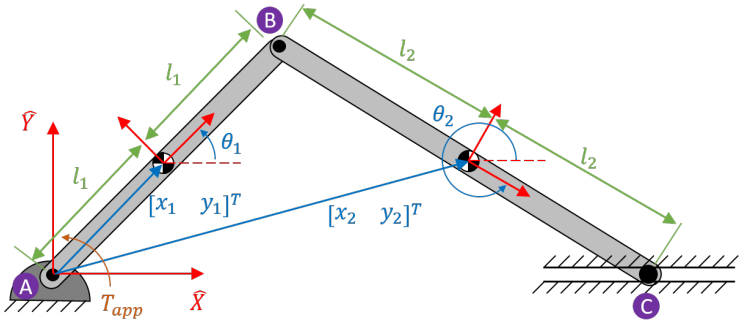

Figure 3.3: Two link slider-crank mechanism [45]

In [57] a two link open kinematic chain with its lengths as random variables is analyzed both kinematically and dynamically. The slider-crank developed for this example uses the same parameters as the two link manipulator, except the link lengths are not random, and only the dynamics is considered. The two link manipulator in [57] is a two DOF system, but the slider-crank is a one DOF system.

3.7.1 Deterministic Slider-Crank Problem Statement

The slider-crank is shown in Fig. 3.3. The link lengths are l_1 and l_2. Points *A* and *B* are revolute joints, and Point *C* is constrained to move in a slot. The mechanism moves in a plane perpendicular to the gravity vector. The mass of link 1 and link 2 are m_1 and m_2, respectively, and the moment of inertia is assumed to be a slender rod i.e., ($J_i = \frac{1}{3} m_i \bar{l}_i^{\,2}, i = 1, 2$). The system is driven by a torque, T_{app}, at Point *A*. The initial conditions are based on $\theta_{1o} \equiv \frac{\pi}{4}$ and $\dot{\theta}_{1o} \equiv 0$.

3.7.2 Inputs to the System

- $\underline{g} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$,
- $n_{\text{dof}} = 4$,
- $n_{\text{bod}} = 2$, and
- $n_{\text{con}} = 5$.

The constraints for the system are two revolute joints and one dot-2 constraint. Recall, the dot-2 constraint ensures that the translation along an axis remains perpendicular to an axis on the body. The revolute joint constraints at Point A are

$$\underline{\Phi}_A = \begin{bmatrix} x_1 - \left(2\, e0_1{}^2 - 1\right) l_1 \\ \\ -2\, e0_1\, e3_1\, l_1 + y_1 \end{bmatrix}. \tag{3.51}$$

The revolute joint constraints for Point B are

$$\underline{\Phi}_B = \begin{bmatrix} x_2 - \left(2\, e0_2{}^2 - 1\right) l_2 - x_1 - \left(2\, e0_1{}^2 - 1\right) l_1 \\ \\ -2\, e0_1\, e3_1\, l_1 - 2\, e0_2\, e3_2\, l_2 - y_1 + y_2 \end{bmatrix}. \tag{3.52}$$

The dot-2 constraint for Point C is

$$\underline{\Phi}_C = -2\, e0_2\, e3_2\, l_2 - y_2. \tag{3.53}$$

The final constraint equation vector with the unit quaternion constraints, that are automatically added by Maple, is

$$\underline{\Phi} = \begin{bmatrix} x_1 - \left(2\, e0_1{}^2 - 1\right) l_1 \\ -2\, e0_1\, e3_1\, l_1 + y_1 \\ x_2 - \left(2\, e0_2{}^2 - 1\right) l_2 - x_1 - \left(2\, e0_1{}^2 - 1\right) l_1 \\ -2\, e0_1\, e3_1\, l_1 - 2\, e0_2\, e3_2\, l_2 - y_1 + y_2 \\ -2\, e0_2\, e3_2\, l_2 - y_2 \\ e0_1{}^2 + e3_1{}^2 - 1 \\ e0_2{}^2 + e3_2{}^2 - 1 \end{bmatrix} . \tag{3.54}$$

The next step is to define the angular velocity for the applied moments of the system. The applied moment is Eqn. (5.1), and the angular velocity of body 1 is

$$\omega = 2\, e0_1\, \dot{e3}_1 - 2\, e3_1\, \dot{e0}_1 \, . \tag{3.55}$$

The last step is to define the elastic potential of the system,

$$V_{\text{elastic}} = 0 \, . \tag{3.56}$$

3.7.3 Maple Automation

Based on Eqn. (3.54), the constraint Jacobian is

$$\mathbf{\Phi}_q = \begin{bmatrix} 1 & 0 & -4\,e0_1\,l_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2\,e3_1\,l_1 & -2\,e0_1\,l_1 & 0 & 0 & 0 & 0 \\ -1 & 0 & -4\,e0_1\,l_1 & 0 & 1 & 0 & -4\,e0_2\,l_2 & 0 \\ 0 & -1 & -2\,e3_1\,l_1 & -2\,e0_1\,l_1 & 0 & 1 & -2\,e3_2\,l_2 & -2\,e0_2\,l_2 \\ 0 & 0 & 0 & 0 & 0 & -1 & -2\,e3_2\,l_2 & -2\,e0_2\,l_2 \\ 0 & 0 & 2\,e0_1 & 2\,e3_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2\,e0_2 & 2\,e3_2 \end{bmatrix}, \tag{3.57}$$

and the nonlinear acceleration terms are

$$\underline{\gamma} = \begin{bmatrix} 4\,\dot{e0}_1^2 l_1 \\ 4\,\dot{e3}_1\,l_1\,\dot{e0}_1 \\ 4\,\dot{e0}_1^2 l_1 + 4\,\dot{e0}_2^2 l_2 \\ 4\,\dot{e3}_1\,l_1\,\dot{e0}_1 + 4\,\dot{e3}_2\,l_2\,\dot{e0}_2 \\ 4\,\dot{e3}_2\,l_2\,\dot{e0}_2 \\ -2\,\dot{e0}_1^2 - 2\,\dot{e3}_1^2 \\ -2\,\dot{e0}_2^2 - 2\,\dot{e3}_2^2 \end{bmatrix}. \tag{3.58}$$

which are automatically calculated by Maple.

Next, the kinetic energy and the potential energy need to be determined to define the system in Eqn. (3.39). The kinetic energy, T, for the system is calculated automatically,

$$\begin{aligned} T =& \frac{1}{2}\left(\dot{x}_1^2 + \dot{y}_1^2\right) m_1 + \left(2\,e0_1{}^2\dot{e3}_1^2 - 4\,e3_1\,\dot{e0}_1\,e0_1\,\dot{e3}_1 + 2\,e3_1{}^2\dot{e0}_1^2\right) J_1 \\ &+ \frac{1}{2}\left(\dot{x}_2^2 + \dot{y}_2^2\right) m_2 + \left(2\,e0_2{}^2\dot{e3}_2^2 - 4\,e3_2\,\dot{e0}_2\,e0_2\,\dot{e3}_2 + 2\,e3_2{}^2\dot{e0}_2^2\right) J_2\,. \end{aligned} \tag{3.59}$$

When applying Lagrange's method, the kinetic energy leads to $\mathbf{M}$, the inertia matrix, and $\underline{C}$ the nonlinear acceleration terms vector. The potential energy, V, for the system is

$$V = 0. \tag{3.60}$$

Applying the Principle of Virtual Work in Eqn. (3.49) with the pre-existing automation scripts

$$\delta\text{Work} = T_{app}\left(2\, e0_1\, \delta e3_1 - 2\, e3_1\, \delta e0_1\right) \ , \tag{3.61}$$

leads to the generalized forces

$$\underline{Q} = \begin{bmatrix} 0 \\ 0 \\ -2\, T_{app} \cdot e3_1 \\ 2\, T_{app} \cdot e0_1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} . \tag{3.62}$$

Applying the Constrained Lagrangian method in the Maple script leads to all of the matrices and vectors in Eqn. (3.39) being defined. The state space system contains sixteen ordinary differential equations, seven position constraints, and seven velocity constraints, for a total of thirty equations of motion.

The Matlab automation part of DMBoD for the deterministic slider-crank is

presented in Section 5.1. The system is simulated numerically in a Monte Carlo analysis and compared to the PCT solution of the same problem.

3.8 Summary of the Multi-Body Dynamic Automation Process

The DMBoD process is not entirely a novel one. Other COTS programs perform the same calculations, albeit not each time the problem is solved. COTS programs can solve for the components of Eqn. (3.39) ahead of time because the simulations are deterministic with a fixed set of generalized coordinates per body. Because of the PCT implementation in the next chapter, the generalized coordinates per body is not fixed. Therefore, the components in Eqn. (3.39) need to be found for each problem.

Also, the use of quaternions for a 3D problem is not novel; however, it is novel for a 2D problem. For a deterministic system, the use of quaternions for a 2D system unnecessarily overcomplicates the derivation. There is an added abstraction on the 2D rotation using quaternions because it results in quadratic terms and not trigonometric. However, this is advantageous when applying PCT to the process, which can be seen in the next chapter.

In this chapter, a general repeatable method has been derived for building the DAE equations of motion of an MBD system and solving them. The method is automated, the DMBoD process, limiting the input the user needs to provide. A slider-crank with applied

torque to the crank is used as an example to show the method and process. Using the DMBoD process as a starting point, the PCT method from Section 2.5 is able to be embedded. A new process is created to build and solve a PCT MBD problem.

CHAPTER 4

Polynomial Chaos Theory Method Applied to A Multi-Body Dynamic System

To integrate the Polynomial Chaos Theory (PCT) method described in Section 2.5 into the Multi-Body Dynamic (MBD) formulation in Chapter 3, additional theory needs to be developed, the concept of Variational Work. With Variational Work, PCT is applied to the components of the Constrained Lagrangian method in Eqn. (3.35). Using Variational Work leads to a more streamlined automation than if PCT were applied directly to the system in Eqn. (3.39).

This chapter derives the theory for Variational Work. The new process using Variational Work is contrasted with the Traditional Process of using PCT on the final state equations using a mass-spring-damper system. Then the automation process in Section 3.6 is updated to use Variational Work. Finally, the derived PCT system using Variational Work is compared to the Monte Carlo method using the slider-crank example.

4.1 Polynomial Chaos Theory Work

For an MBD system, some of the complexity of the Galerkin projections can be reduced or eliminated. Variational Work can be applied only to those terms in Eqn. (3.35) that are defined using energy or the Principle of Virtual Work. Unfortunately, there is nothing that reduces the number of inner products required to project the random variables

onto the orthogonal basis set or the kinematic constraints (see Section 2.5.2 for review of inner products). Projecting the components required to formulate the Constrained Lagrangian onto the orthogonal polynomial basis via the Galerkin projections can lead to a reduction in the complexity of the calculation. Reduction in the complexity can help speed up the Maple processing to get a system even faster. In theory, given a large enough system, there would not be enough computing power to handle the Galerkin projections if they were performed on the final equations of motion.

To use the Constrained Lagrangian in Eqn. (3.35), the concept of work with variation needs to be expanded. Specifically, kinetic and potential energy and the Principle of Virtual Work need to be derived. Work is the product of the force over a distance, and work is a scalar not a vector. In terms of vector spaces,

$$\text{Work} = \langle \underline{F}\left(\underline{r}\right), \underline{r} \rangle \,, \tag{4.1}$$

where "$\langle\rangle$" is the inner product, $\underline{F}$ is an applied force, and $\underline{r}$ is the distance the force acted. For Cartesian space, the inner product is the dot product,

$$\text{Work} = \int \underline{F}\left(\underline{r}\right) \bullet d\underline{r} \,. \tag{4.2}$$

For an orthogonal polynomial basis, the inner product is defined in Eqn. (2.19). If $\underline{F}\left(\underline{r}\right)$

and $\underline{r}$ are functions of the random variable, $\underline{\xi}$, then two inner products need to be performed; one for Cartesian space and one for the orthogonal polynomial basis.

If it is assumed that $\underline{F}$ is not a function of $\underline{r}$, then Eqn. (4.2) can be written as

$$\text{Work} = \underline{F} \bullet \underline{r} \, . \tag{4.3}$$

It is very common in mechanical systems for $\underline{F}$ to not be a function of $\underline{r}$; this occurs in non-conservative forces.

To compute the variational work, $\underline{F}$ and $\underline{r}$ are first projected onto the orthogonal polynomial basis using Eqn. (2.19). This is the first inner product. Then using Eqn. (4.3),

$$\overset{\circ}{\text{Work}} = \underline{\overset{\circ}{F}} \bullet \underline{\overset{\circ}{r}} \, , \tag{4.4}$$

where "$\overset{\circ}{\bullet}$" denotes that the variable is projected onto the orthogonal polynomial basis. Equation (4.4) is used in the Variational Principle of Virtual Work.

Another approach to compute work with variation is to compute the work in Cartesian space and then use the definition of the inner product for the orthogonal polynomial basis, Eqn. (2.19). This leads to the calculation of work

$$\overset{\circ}{\text{Work}} = \int \left(\int \underline{F} \left(\underline{r}, \underline{\xi} \right) d\underline{r} \left(\underline{\xi} \right) \right) W \left(\underline{\xi} \right) d\underline{\xi} \, . \tag{4.5}$$

$\mathring{\text{Work}}$ can be defined further as

$$\mathring{\text{Work}} = \int \text{Work}\left(\underline{r}, \underline{\xi}\right) W\left(\underline{\xi}\right) d\underline{\xi} \,. \tag{4.6}$$

Equation (4.6) allows classical equations of energy for a system to be projected onto the orthogonal polynomial basis. Without Eqn. (4.6), the Constrained Lagrangian would not be as advantageous to use.

Both Eqns. (4.4) and (4.6) form the basis for Variational Work. With these two versions of Variational Work, the Constrained Lagrangian method for deriving the equations of motion can be used. Without Variational Work, PCT would be applied to the final equations of motion. This would lead to very complex coefficient calculation while projecting the equations of motion onto the orthogonal polynomial basis. With Variational Work, PCT application can be compartmentalized and smaller, less complex Galerkin projections can be performed.

4.1.1 Polynomial Chaos Theory Energy

For a mechanical system, there are three common types of work: kinetic energy, gravitational potential and elastic potential. All of these energies can be handled the same way when projecting onto the orthogonal polynomial basis.

The kinetic energy, gravitational energy, and elastic potential energy for a

mechanical system are derived in Eqns. (3.43), (3.47), and (3.42), respectively. Equation (4.6) is used to project these onto the orthogonal polynomial basis

$$\mathring{T} = \int T\left(\underline{\xi}\right) W\left(\underline{\xi}\right) d\underline{\xi} \,, \tag{4.7}$$

$$\mathring{V}_{\text{grav}} = \int V_{\text{grav}}\left(\underline{\xi}\right) W\left(\underline{\xi}\right) d\underline{\xi} \,, \text{ and} \tag{4.8}$$

$$\mathring{V}_{\text{elastic}} = \int V_{\text{elastic}}\left(\underline{\xi}\right) W\left(\underline{\xi}\right) d\underline{\xi} \,. \tag{4.9}$$

The total potential energy for the system to be used in Eqn. (3.35) is

$$\mathring{V} = \mathring{V}_{\text{grav}} + \mathring{V}_{\text{elastic}} \,. \tag{4.10}$$

4.1.2 Polynomial Chaos Theory Principle of Virtual Work

With the kinetic and potential energy projected onto the orthogonal polynomial basis, the final piece of Eqn. (3.35) is to address computation of the generalized forces, Q_i. To find the generalized forces in Eqn. (3.35), Q_i, the Principle of Virtual Work, needs to be projected onto the orthogonal polynomial basis. The Principle of Virtual Work is used to find the generalized forces resulting from conservative nonlinear forces or non-conservative applied forces. The Principle of Virtual Work for a mechanical system is defined in Eqn. (3.49). Because $\underline{F}_i$ and $\underline{M}_j$ in Eqn. (3.49) are arbitrary and can be nonlinear, the easiest way to project these forces and moments onto the orthogonal

polynomial basis is to use the first method for projecting Work, Eqn. (4.4)

$$\delta \overset{\circ}{W}ork = \sum_{i=1}^{n_{\text{frc}}} \underline{\overset{\circ}{F}}_i \cdot \delta \underline{\overset{\circ}{r}}_i + \sum_{j=1}^{n_{\text{mom}}} \underline{\overset{\circ}{M}}_j \cdot \delta \underline{\overset{\circ}{\theta}}_j \ . \tag{4.11}$$

This form of the Variational Work can be used because the applied forces and moments are never a function of the virtual displacements.

PCT Work and PCT Principle of Virtual work are the final pieces needed to embed PCT into the Constrained Lagrangian method. Now a process can be discussed that incorporates Variational Work.

4.1.3 Variational Work Process

The Variational Work Process applies PCT to the components of Eqn. (3.35), then uses the Constrained Lagrangian method to derive the equations of motion. To project the various components or equations onto the orthogonal polynomial basis, a simple algorithm is needed. Algorithm 1 loops through each component or equation and projects it onto the orthogonal polynomial basis component via the Galerkin projection in Eqn. (2.19).

To help explain the Variational Work Process, it will be contrasted with the Traditional Process. The Traditional Process is the typical method used, applying PCT to the final equations of motion of the system.

Algorithm 1 Equation PCT Projection

procedure GALERKINPROJECTION(f, Ψ)
  $n \leftarrow$ *number of equations in* Ψ
  $i = 0$
  for *each equation in* f **do**
    $i = i + 1$
    $j = 0$
    for *each* Ψ **do**
      $j = j + 1$
      $index = n * (i - 1) + j$
      $\mathring{f}\,(index) = \langle f\,(i)\,, \Psi\,(j) \rangle$
    end for
  end for
  return $\mathring{f}$
end procedure

Traditional Process

The Traditional Process derives the equations of motion for the system resulting in an index-3 differential algebraic equation (DAE) system. PCT is applied then to the index-3 DAEs [48, 49]. To give as close of a comparison to the Variational Work Process, the equations of motion in the Traditional Process are derived using a Constrained Lagrangian method. In theory, since the Traditional Process is applied to the final equations of motion, any MBD method can be used. The Traditional Process is:

1. Derive $\underline{\Phi} = \underline{0}$
2. Calculate constraint Jacobian, $\mathbf{\Phi}_q$
3. Calculate the energy for the system

- Calculate T, Eqn. (3.43)
- Calculate V_{grav}, Eqn. (3.47)
- Calculate V_{elastic}, Eqn. (3.42)

4. Use the Principle of Virtual work to compute generalized forces
   - Calculate $\sum_{i=1}^{n_{\text{frc}}} \underline{F}_i$ and $\sum_{i=1}^{n_{\text{mom}}} \underline{M}_i$
   - Calculate $\sum_{i=1}^{n_{\text{frc}}} \delta \underline{r}_i$ and $\sum_{i=1}^{n_{\text{mom}}} \delta \underline{\theta}_i$
   - Calculate Virtual Work, Eqn. (3.49)
   - Calculate $\underline{Q}$, Eqn. (4.65)
5. Perform Constrained Lagrangian operations
6. Compute a_i, Eqn. (2.18) and substitute into system in Eqn. (3.1)
7. Expand $\underline{q}$ and $\underline{\lambda}$ in terms of the orthogonal polynomial basis using Eqn. (2.16)
8. Substitute into system
9. For each equation of the DAE, use Algorithm 1

Using these steps, the PCT equations of motion are derived. In general, using Algorithm 1 on the final equations of motion results in more complicated Galerkin projections. These complicated projections can result in equations that are too complicated for a computer to solve or to simplify. Not being able to simplify the equations can result in less efficient equations of motion, thus, making the cycle time longer than necessary.

Variational Work

The Variational Work Process must use an energy-based MBD formulation. The Constrained Lagrangian method is chosen for building the equations of motion. The Variational Work Process also results in an index-3 DAE. The Variational Work Process is:

1. Derive $\underline{\Phi} = \underline{0}$
2. Compute a_i, Eqn. (2.18) and expand random variables
3. Expand $\underline{q}$ and $\underline{\lambda}$ in terms of the orthogonal polynomial basis using Eqn. (2.16)
4. Calculate PCT $\underline{q}$ and $\underline{\lambda}$ using Algorithm 1
5. Calculate PCT kinematics
    - Calculate PCT $\overset{\circ}{\underline{\Phi}}$ using Algorithm 1
    - Calculate PCT $\overset{\circ}{\mathbf{\Phi}}_q$
6. Calculate Variational Energy
    - Calculate T, Eqn. (3.43)
    - Calculate $\overset{\circ}{T}$, Eqn. (4.7)
    - Calculate V_{grav}, Eqn. (3.47)
    - Calculate $\overset{\circ}{V}_{\text{grav}}$, Eqn. (4.8)
    - Calculate V_{elastic}, Eqn. (3.42)

- Calculate $\mathring{V}_{\text{elastic}}$, Eqn. (4.9)
- Calculate $\mathring{V}$, Eqn. (4.10)

7. Calculate PCT Generalized Forces using the Variational Principle of Virtual Work
   - Calculate $\sum_{i=1}^{n_{\text{frc}}} \underline{\mathring{F}}_i$ and $\sum_{i=1}^{n_{\text{mom}}} \underline{\mathring{M}}_i$, Algorithm 1
   - Calculate $\sum_{i=1}^{n_{\text{frc}}} \delta\underline{\mathring{r}}_i$ and $\sum_{i=1}^{n_{\text{mom}}} \delta\underline{\mathring{\theta}}_i$, Algorithm 1
   - Calculate Variational Virtual Work, Eqn. (4.11)
   - Calculate $\underline{\mathring{Q}}$, Eqn. (4.65)
8. Perform Constrained Lagrangian operations

The Variational Work Process applies PCT throughout the whole process in intermediate steps. This results in simpler Galerkin projections, but possibly more of them. This leads to a faster algorithm building the final equations of motion, and possibly more efficient equations because they may be easier to simplify.

4.1.4 Example: Mass-Spring-Damper

A mass-spring-damper is used to contrast the use of the Variational Work Process with the Traditional Process of applying PCT to the deterministic equations of motion. This is a simple example, but it highlights the differences between the two methods.

The mass-spring-damper is shown in Fig. 4.1. Variability is assumed to exist in the

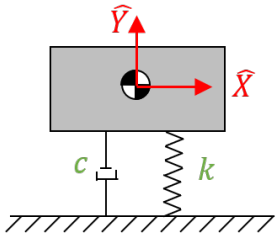

Figure 4.1: Mass-Spring-Damper

spring stiffness, k, with a normal distribution: mean of $\bar{k}$ and standard deviation of σ_k. The system is represented with two generalized coordinates $q = \begin{bmatrix} x & y \end{bmatrix}^T$. Gravity is acting in the $-\hat{Y}$ direction. The mass is constrained to move only along the $\hat{Y}$ axis.

Hermite Polynomials

There is overlap in some of the calculations between the Traditional Process and Variational Work Process. One of the main overlaps is the projection of the random variable onto the orthogonal polynomial basis. Hermite Polynomials are the best choice for the PCT expansion of spring stiffness because it is normally distributed (see Tab. 2.1). There are two weighting functions commonly used with Hermite Polynomials; for this example, the probability weighting function is used [63]:

$$w\left(\xi\right) = \frac{1}{\sqrt{2\pi}} e^{\frac{-\xi^2}{2}} \ . \tag{4.12}$$

With this weighting function, the first three terms of the corresponding Hermite PCT basis set are

$$\underline{H}\left(\xi\right) = \begin{bmatrix} 1 \\ \xi \\ \xi^2 - 1 \end{bmatrix} . \tag{4.13}$$

Substituting Eqn. (4.12) and the basis functions in Eqn. (4.13) into Eqn. (2.14) yields

$$\frac{1}{\sqrt{2\pi}} \int H_i\left(\xi\right) H_j\left(\xi\right) e^{\frac{-\xi^2}{2}} d\xi = n! \delta_{ij} \, . \tag{4.14}$$

The stiffness, k, is expanded using Eqn. (2.16) and yields

$$k \approx \sum_{j=0}^{p} a_j H_j\left(\xi\right) \, . \tag{4.15}$$

To determine the coefficients a_j in Eqn. (4.15), they need to be projected onto the Hermite Polynomial basis set [57]

$$a_j = \frac{\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \left(\bar{k} + \sigma_k \eta\right) H_i\left(\eta\right) e^{\frac{-\eta^2}{2}} d\eta}{\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} H_j\left(\eta\right) H_i\left(\eta\right) e^{\frac{-\eta^2}{2}} d\eta} \, . \tag{4.16}$$

Using Eqn. (4.14) to simplify the expression,

$$a_j = \frac{1}{n!\sqrt{2\pi}} \int_{-\infty}^{\infty} \left(\bar{k} + \sigma_k \eta\right) H_i\left(\eta\right) e^{\frac{-\eta^2}{2}} d\eta \, . \tag{4.17}$$

Finding each coefficient in Eqn. (4.15), for $j = 0$,

$$a_0 = \frac{1}{0!\sqrt{2\pi}} \int_{-\infty}^{\infty} \left(\bar{k} + \sigma_k \eta\right) (1)\, e^{\frac{-\eta^2}{2}} d\eta = \bar{k} \ , \tag{4.18}$$

for $j = 1$,

$$a_1 = \frac{1}{1!\sqrt{2\pi}} \int_{-\infty}^{\infty} \left(\bar{l}_i + \sigma_{li} \eta\right) (\eta)\, e^{\frac{-\eta^2}{2}} d\eta = \sigma_k \ . \tag{4.19}$$

and for $j = 2$,

$$a_2 = \frac{1}{2!\sqrt{2\pi}} \int_{-\infty}^{\infty} \left(\bar{k} + \sigma_k \eta\right) \left(\eta^2 - 1\right) e^{\frac{-\eta^2}{2}} d\eta = 0 \ . \tag{4.20}$$

All coefficients when $j \geq 2$ are zero. Using the calculated coefficients, a_j, for Eqn. (4.15), the PCT expansion of the stiffness k is exact,

$$k = \bar{k} + \sigma_k \xi \ . \tag{4.21}$$

All of the states of the model are expanded using the first two terms of Eqn. (4.13). Expanding them based on these two orthogonal polynomial vectors assumes their responses are normally distributed as well.

Traditional Process

Using the steps for the Traditional Process, the PCT equations of motion are derived for the mass-spring-damper. The following are the results of each step. Because

the only constraint in the system is to ensure there is no motion in the $\hat{X}$ direction, Steps 1 and 2 result in kinematic constraints

$$\Phi = x = 0 \ , \tag{4.22}$$

and constraint Jacobian

$$\mathbf{\Phi}_q = \begin{bmatrix} 1 & 0 \end{bmatrix} \ . \tag{4.23}$$

In Step 3, the energy for the system is built,

$$T = \frac{1}{2} m \left(\dot{x}^2 + \dot{y}^2 \right) \ , \tag{4.24}$$

$$V_{grav} = mgy \ , \text{ and} \tag{4.25}$$

$$V_{elastic} = \frac{1}{2} k y^2 \ . \tag{4.26}$$

Next, in Step 4 the Principle of Virtual Work is applied to calculate the generalized forces. The applied force is

$$F = -c\dot{y} \ , \tag{4.27}$$

and the corresponding virtual displacement is

$$\delta r = \delta y \ . \tag{4.28}$$

The virtual work is then

$$\delta \text{Work} = -c\dot{y}\,\delta y\ , \tag{4.29}$$

which yields the generalized force vector

$$\underline{Q} = \begin{bmatrix} 0 \\ \\ -c\dot{y} \end{bmatrix}\ . \tag{4.30}$$

Using the Constrained Lagrangian formulation in Step 5, the deterministic equations of motion are derived:

$$m\ddot{x} = \lambda\ , \tag{4.31}$$

$$m\ddot{y} + ky + mg = -c\dot{y}\ , \text{ and} \tag{4.32}$$

$$x = 0\ . \tag{4.33}$$

Up until this point, PCT is not included. Projecting the random variable and expanding the generalized coordinates and Lagrange Multipliers in Steps 6 and 7 yields

$$k = \bar{k} + \sigma_k \xi\ , \tag{4.34}$$

$$x = x_0 + x_1 \xi\ , \tag{4.35}$$

$$y = y_0 + y_1 \xi\ , \text{ and} \tag{4.36}$$

$$\lambda = \lambda_0 + \lambda_1 \xi\ . \tag{4.37}$$

The generalized coordinates and Lagrange Multipliers are assumed to be normally distributed, which is a very good assumption in this particular case. Substituting in the random variable and generalized coordinates and Lagrange Multipliers expansions into the equations of motion in Step 8 yields the PCT embedded equations of motion

$$m\left(\ddot{x}_0 + \ddot{x}_1\xi\right) = \lambda_0 + \lambda_1\xi \ , \tag{4.38}$$

$$m\left(\ddot{y}_0 + \ddot{y}_1\xi\right) + \left(\bar{k} + \sigma_k\xi\right)\left(y_0 + y_1\xi\right) + mg = -C\left(\dot{y}_0 + \dot{y}_1\xi\right) \ , \text{ and} \tag{4.39}$$

$$x_0 + x_1\xi = 0 \ . \tag{4.40}$$

Using Algorithm 1 in the final step yields

$$m\ddot{x}_0 = \lambda_0 \ , \tag{4.41}$$

$$m\ddot{x}_1 = \lambda_1 \ , \tag{4.42}$$

$$m\ddot{y}_0 + \bar{k}y_0 + \sigma_k y_1 + mg = -c\dot{y}_0 \ , \tag{4.43}$$

$$m\ddot{y}_1 + \sigma_k y_0 + \bar{k}y_1 = -c\dot{y}_1 \ , \tag{4.44}$$

$$x_0 = 0 \ , \text{ and} \ , \tag{4.45}$$

$$x_1 = 0 \ . \tag{4.46}$$

The PCT method is added in the last couple steps. The full system of equations of the deterministic model is derived before even implementing PCT. Typically, although not

in this case, the ordinary differential equations (ODE) are a lot more complicated, and applying PCT is more difficult.

Variational Work Process

Using the Variational Work Process on the mass-spring-damper, the following are the results of each step. Step 1 results in kinematic constraints, which is the same as Eqn. (4.22). The random variable is projected, and the generalized coordinates and Lagrange Multipliers are expanded in Step 2 - 3 yielding the Eqns. (4.34) - (4.37). Step 4 projects the generalized coordinates of the system

$$\underset{\circ}{\underline{q}} = \begin{bmatrix} x_0 \\ x_1 \\ y_0 \\ y1 \end{bmatrix} \tag{4.47}$$

and the Lagrange Multipliers

$$\underset{\circ}{\underline{\lambda}} = \begin{bmatrix} \lambda_0 \\ \lambda_1 \end{bmatrix} . \tag{4.48}$$

Step 5 projects the kinematic constraint equations and constraint Jacobian onto the orthogonal polynomial basis, embedding PCT into the process earlier, yielding

$$\underline{\mathring{\Phi}} = \begin{bmatrix} x_0 \\ \\ x_1 \end{bmatrix} = \underline{0} \tag{4.49}$$

and the PCT constraint Jacobian

$$\mathring{\mathbf{\Phi}}_q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \\ 0 & 1 & 0 & 0 \end{bmatrix} . \tag{4.50}$$

The energy of the system and the Variational Energy are calculated in Step 6. The kinetic energy is the same as Eqn. (4.24). The variational kinetic energy is

$$\mathring{T} = \frac{1}{2} m \left(\dot{x}_0^2 + \dot{x}_1^2 + \dot{y}_0^2 + \dot{y}_1^2 \right) . \tag{4.51}$$

The gravitational potential energy of the system is calculated, which is the same as Eqn. (4.25). The variational gravitational potential energy is

$$\mathring{V}_{\text{grav}} = m g y_0 . \tag{4.52}$$

The elastic potential energy of the system is calculated, which is the same as Eqn. (4.25).

The variational elastic potential energy is

$$\mathring{V}_{\text{elastic}} = \frac{1}{2}\bar{k}y_0^2 + \bar{k}y_1^2 + \sigma_k y_0 y_1 \ . \tag{4.53}$$

This leads to the total PCT potential energy,

$$\mathring{V} = mgy_0 + \frac{1}{2}\bar{k}y_0^2 + \bar{k}y_1^2 + \sigma_k y_0 y_1 \ . \tag{4.54}$$

In Step 7, the PCT generalized forces are calculated using the Variational Principle of Virtual work. The applied force projected onto the orthogonal polynomial basis is

$$\mathring{F} = -c \begin{bmatrix} \dot{y}_0 \\ \\ \dot{y}_1 \end{bmatrix} \ . \tag{4.55}$$

The corresponding virtual displacement is

$$\delta\mathring{r} = \begin{bmatrix} \delta y_0 \\ \\ \delta y_1 \end{bmatrix} \ . \tag{4.56}$$

With $\mathring{F}$ and $\delta\mathring{r}$ defined, the Variational Principle of Virtual Work is

$$\delta\mathring{\text{Work}} = -c\dot{y}_0\,\delta y_0 - c\dot{y}_1\,\delta y_1 \ , \tag{4.57}$$

which yields the generalized forces

$$\mathring{Q} = \begin{bmatrix} 0 \\ 0 \\ -c\dot{y}_0 \\ -c\dot{y}_1 \end{bmatrix} . \tag{4.58}$$

Using the Constrained Lagrangian formulation in Step 16,

$$m\ddot{x}_0 = \lambda_0 \ , \tag{4.59}$$

$$m\ddot{x}_1 = \lambda_1 \ , \tag{4.60}$$

$$m\ddot{y}_0 + +\bar{k}y_0 + \sigma_k y_1 + mg = -c\dot{y}_0 \ , \tag{4.61}$$

$$m\ddot{y}_1 + \sigma_k y_0 + \bar{k}y_1 = -c\dot{y}_1 \ , \tag{4.62}$$

$$x_0 = 0 \ , \text{ and} \tag{4.63}$$

$$x_1 = 0 \ . \tag{4.64}$$

Using both the Traditional Process and the Variational Work Process yields the same equations of motion. The Traditional Process applies the Galerkin projections to the final system of equations. The Variational Work Process applies the Galerkin projections to each of the components, resulting in simpler calculations. If the system grows and generalized coordinates are not as easily decoupled, especially when 3D rotations are

involved, projecting the equations of motion is more difficult than projecting the energy. Embedding PCT earlier into the derivation process lends itself to be better for automation. The calculations are not as difficult acting on position and velocity terms rather than on acceleration terms in the Traditional Process. The Variational Work Process is able to run faster due to the less complicated projections. Also, with less complicated projections, the calculations can be simplified easier, compared to performing the equations on acceleration level equations.

4.2 Automation

With the process using Variational Work defined, it is now appropriate for it to be placed into DMBoD defined in Section 3.6. The flow chart for the Polynomial Chaos Multi-Body Dynamics (PCMBoD) automation process, the automation of the PCT equations of motion and their solution, can be seen in Fig. (4.2). Similarly to the deterministic process shown in Fig. 3.2, PCMBoD uses the same inputs with the addition of the random variables. The major addition is that the Variational Work Process is applied in Maple. The process starts with Maple to find symbolically each of the matrices and vectors needed to solve Eqn. (3.39). These matrices and vectors are exported as Matlab functions. In Matlab, the initial conditions are specified, and the system is simulated.

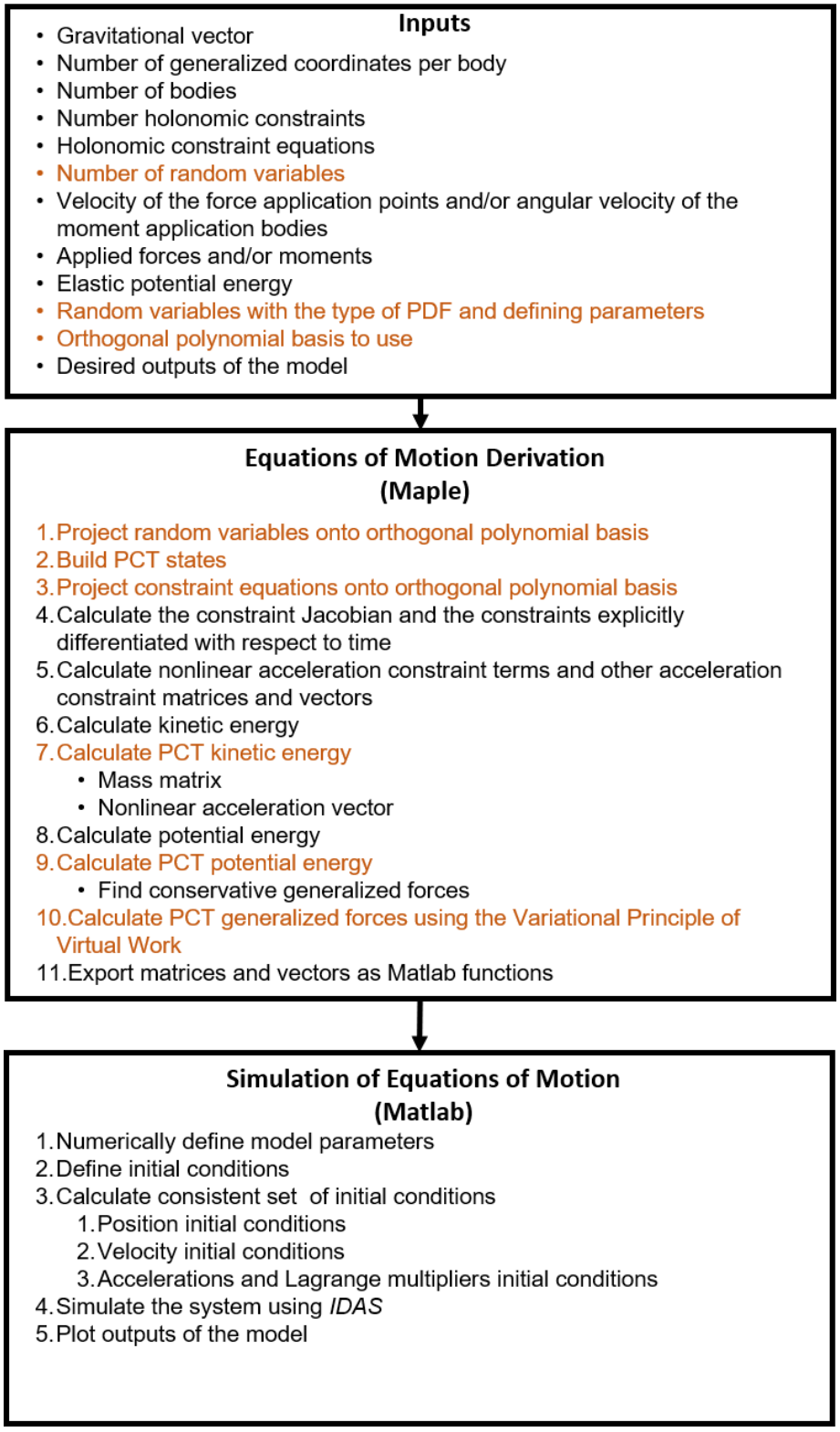

Figure 4.2: PCMBoD process flow chart where the orange font shows the changes from the DMBoD process

4.2.1 Inputs

To set up the dimensionality of the PCT system, additional information is needed. The PCT system has the same inputs as the deterministic system with the addition of the number of random variables, n_{pct}.

In addition to using the holonomic constraint definitions in Section 3.3, the constraint vector $\underline{\Phi}$ is defined. Each of the applied forces and moments is defined along with the corresponding velocity variable of the point for a force or the angular velocity variable of the body for a moment. Then the elastic potential energy is defined. The random variables need to be defined with the corresponding probability density function (PDF) and defining parameters. Next, the orthogonal polynomial basis needs to be selected; this should be based on the best selection in Table 2.1. With the orthogonal polynomial basis selected and the random variables fully defined, PCMBoD can build PCT vectors, $\underline{\Psi}$. Finally, similar to the deterministic process the outputs for the model are defined.

4.2.2 Maple Automation Subprocess

With the dimensionality of the system defined, the kinematic constraints defined and the forces and moments and the velocity and angular velocity of their application

defined, the system, the components of Eqn. (3.39) can be calculated. This can be seen in the following procedure, which follows very closely the procedure in Section 3.6.2.

First, the random variables are expanded via the orthogonal polynomial basis, and the coefficients are calculated. The generalized coordinates and Lagrange Multipliers are expanded next. The PCT generalized coordinates and the Lagrange Multipliers are computed using Algorithm 1.

Next, the kinematic constraints are projected using Algorithm 1. Then the constraint Jacobian, $\mathring{\mathbf{\Phi}}_q$, and the constraint vector differentiated explicitly with respect to time, $\underline{\mathring{\Phi}}_t$, are calculated. Next, the PCT acceleration components of the constraint equations are found.

The energy of the system is the next step in the automation. The kinetic energy of the system is calculated in Eqn. (3.43). Using Eqn. (4.7), the variational kinetic energy is calculated. Applying the Constrained Lagrangian leads to $\mathbf{M}$ and $\underline{C}$ in Eqn. (3.39) by Eqn. (3.45) and Eqn. (3.46), respectively, using the PCT generalized coordinates, $\underline{\mathring{q}}$.

The potential energy for the system contains two components, the gravitational potential and the elastic potential. The gravitational potential is calculated in Eqn. (3.47). The variational gravitational potential is calculated in (4.8). The elastic potential energy was defined as in input, and the PCT potential energy is calculated using (4.9). The total potential energy is Eqn. (4.10).

Finally, the generalized forces are found with help from the Variational Principle of Virtual Work. The applied forces, moments and virtual displacements are projected using Algorithm 1. The Variational Virtual Work is calculated then by Eqn. (4.11).

To find the generalized forces, the potential energy in Eqn. (4.10) and Eqn. (4.11) are used. The generalized force for PCT generalized coordinate *k* is

$$\mathring{Q}_k = \frac{\partial \left(\delta \mathring{\mathrm{Work}}\right)}{\partial \left(\delta \mathring{q}_k\right)} - \frac{\partial \mathring{V}}{\partial \mathring{q}_k} \,. \tag{4.65}$$

Using the Constrained Lagrangian in Eqn. (3.35), the various matrices and vectors are calculated for Eqn. (3.39). These components are exported as Matlab functions. The system in Eqn. (3.39) is built with custom Maple functions. The descriptions of the functions can be found in Appendix A.

4.2.3 Matlab Automate Subprocess

With the components of the equations exported from Maple as Matlab functions, the final input is the set of initial conditions for the free DOFs of the system. Matlab then solves for a consistent set of initial conditions. Finally, the system is simulated, and the outputs are plotted.

4.3 Example: Polynomial Chaos Theory Slider-Crank

As an example to show how the PCMBoD automation process is applied, the deterministic slider-crank in Section 3.7 is be used. Recall that the slider-crank mechanism being analyzed is based on the work from [57] and has been analyzed before by applying PCT to the equations of motion [45]. The slider-crank problem is updated to add variation to the link lengths.

4.3.1 Problem

The slider-crank is shown in Fig. 3.3. Uncertainty in the link lengths is incorporated by simulating populations of each with normal distributions: means of $\bar{l_1}$ and $\bar{l_2}$ and standard deviations of σ_{l1} and σ_{l2}, respectively. Points *A* and *B* are revolute joints, and Point *C* is constrained to move in a slot. The mechanism moves in a plane perpendicular to the gravity vector. The mass and the moment of inertia are assumed to be constant as the link lengths vary. The system is driven by a torque, T_{app}, at Point *A*.

4.3.2 Inputs

The inputs to the system are

- $\underline{g} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$,

- $n_{\text{dof}} = 4$,
- $n_{\text{bod}} = 2$,
- $n_{\text{con}} = 5$, and
- $n_{\text{pct}} = 2$, because there are two random variables.

The kinematic constraints are exactly the same as the constraints in the deterministic slider-crank in Section 3.7, Eqn. (3.54). The angular velocity of the applied moments of the system is Eqn. (3.55). The elastic potential energy is defined in Eqn. (3.56).

Finally, the last input needed for the PCT automation is the description of the random variables. The random variables in this example for the slider-crank are the link lengths, which are taken from normal distributions with prescribed means and standard deviations. Because the random variables are taken from normal distributions, a Hermite Polynomial the orthogonal basis is used (see Table. 2.1).

4.3.3 Maple Automation

All the calculations in this section are part of the automated process. They are shown only for comparison to the deterministic slider-crank example in Section 3.7.

The first step in the automation is to project the random variables onto the

orthogonal polynomial basis. The weighting functions used is Eqn. (4.12) with the same Hermite PCT basis set as Eqn. (4.13).

The link length, l_i, is expanded using Eqn. (2.16),

$$l_i \approx \sum_{j=0}^{P} a_j \Psi_j(\xi) . \tag{4.66}$$

Using equation Eqn. (2.18), each coefficient is

$$a_0 = \bar{l_i} \text{ , and} \tag{4.67}$$

$$a_1 = \sigma_{li} . \tag{4.68}$$

The PCT expansion of the length l_i is exact for this example. The expansions of each link of the slider-crank are

$$l_1 = \bar{l_1} + \sigma_{l1}\xi_1 \text{ , and} \tag{4.69}$$

$$l_2 = \bar{l_2} + \sigma_{l2}\xi_2 . \tag{4.70}$$

After expanding the random variables, the automation process expands the generalized coordinates, $\underline{q} = \begin{bmatrix} x_1 & y_1 & e0_1 & e3_1 & x_2 & y_2 & e0_2 & e3_2 \end{bmatrix}^T$, and Lagrange multipliers, $\underline{\lambda} = \begin{bmatrix} \lambda_1 & \lambda_2 & \lambda_3 & \lambda_4 & \lambda_5 \end{bmatrix}^T$, in terms of the PCT orthogonal basis. Using Eqn. (2.15) with $n_v = 2$ and $n_p = 1$, the number of PCT terms is 2. Expanding the

generalized coordinates and Lagrange multipliers yields

$$q_i = q_{i0} + q_{i1}\xi_1 + q_{i2}\xi_2 \ , \text{ and} \tag{4.71}$$

$$\lambda_i = \lambda_{i0} + \lambda_{i1}\xi_1 + \lambda_{i2}\xi_2 \ . \tag{4.72}$$

These expansions result in the PCT generalized coordinates for body *i*

$$\mathring{q}_i = \begin{bmatrix} x_{i0} \\ x_{i1} \\ x_{i2} \\ y_{i0} \\ y_{i1} \\ y_{i2} \\ e0_{i0} \\ e0_{i1} \\ e0_{i2} \\ e3_{i0} \\ e3_{i1} \\ e3_{i2} \end{bmatrix} \tag{4.73}$$

and Lagrange multiplier *j*

$$\mathring{\lambda}_j = \begin{bmatrix} \lambda_{j0}(t) \\ \lambda_{j1}(t) \\ \lambda_{j2}(t) \end{bmatrix} . \tag{4.74}$$

With the random variables and generalized coordinates expanded in terms of the orthogonal polynomial basis, they are substituted into the kinematic constraint equations Eqn. (3.54) and the angular velocity of body 1 in Eqn. (3.55). Using Algorithm 1 with the kinematic constraints and the Hermite Polynomial basis in Eqn. (4.13), the kinematic constraints can be projected onto the orthogonal polynomial basis. Using the dot-2 constraint, as an example, at Point C, the resulting constraint equations are

$$\mathring{\Phi}_C = -2 \begin{bmatrix} (e3_{20}\, e0_{20} + e3_{21}\, e0_{21} + e0_{22}\, e3_{22})\, \bar{l_2} \\ + (e3_{21}\, e0_{20} + e3_{20}\, e0_{21})\, \sigma_{l2} + \dfrac{y_{20}}{2} \\ \hdashline (e3_{21}\, e0_{20} + e3_{20}\, e0_{21})\, \bar{l_2} \\ + (e3_{20}\, e0_{20} + 3\, e3_{21}\, e0_{21} + e0_{22}\, e3_{22})\, \sigma_{l2} + \dfrac{y_{21}}{2} \\ \hdashline (e0_{20}\, e3_{22} + e0_{22}\, e3_{20})\, \bar{l_2} \\ + (e3_{22}\, e0_{21} + e3_{21}\, e0_{22})\, \sigma_{l2} + \dfrac{y_{22}}{2} \end{bmatrix} . \tag{4.75}$$

These equations are calculated automatically and only shown for comparison.

Computing the PCT constraint Jacobian and nonlinear acceleration vector are

straightforward. These terms are computed the same as the deterministic case, however the differentiation is done with respect to the generalized coordinates defined in Eqn. (4.73).

The kinetic energy and the potential energy need to be determined to use the Constrained Lagrangian. The kinetic energy, T, is defined in Eqn. (3.59). Using Eqn. (4.7) with Eqn. (3.59), the variational kinetic energy, $\mathring{T}$, can be computed. The potential energy, V, for the system is exactly the same as Eqn. (3.60),

$$V = \mathring{V} = 0 \; . \tag{4.76}$$

Finally, using the Variational Principle of Virtual Work, the applied moment needs to be projected onto the orthogonal polynomial basis using Eqn. (4.11). The applied moment and virtual displacements are projected using Algorithm 1. With the applied moment and virtual displacement projected, the virtual work for the system can be calculated. This process ultimately leads to the generalized force vector, $\underline{\mathring{Q}}$, using Eqn. (4.65).

Using all of the PCT components, $\mathring{T}$, $\mathring{V}$, $\underline{\mathring{Q}}$, $\mathring{\mathbf{\Phi}}_q$, and $\underline{\mathring{\Phi}}$, the PCT equations of motion of the system can be derived. The equations derived are identical to the PCT equations derived by applying Algorithm 1 to the deterministic equations of motion using Maple.

The numerical solution of the system is shown in Chapter 5, where the PCT method is compared to a Monte Carlo Analysis.

4.4 Summary

In this chapter, the concept of Variational Work was derived. A procedure for using Variational Work was outlined and compared to the Traditional Process using a mass-spring-damper example. Next, the automation algorithm was updated to include Variational Work. A slider-crank with applied torque to the crank was used as an example to demonstrate the automation.

Using the Variational Work process embeds PCT into derivation of the equations of motion. This allows for a more integrated automation process, which reduces the complexity of the Galerkin projections. The reduction results in a faster automation process. Also, the reduction can lead to a more efficient solution by allowing the equations to be simplified easier. Embedding PCT earlier into the process enables the full advantages of the Constrained Lagrangian to be realized.

In the next chapter, the slider-crank in Section 3.7 in a Monte Carlo analysis and the slider-crank in this chapter are numerically solved. Additional, more complicated examples are presented and solved using a Monte Carlo analyses and PCT.

CHAPTER 5

Examples Using the Deterministic and Polynomial Chaos Multi-Body Dynamics Automation Processes

This chapter offers several examples showcasing the Deterministic Multi-Body Dynamic Process (DMBoD) and the Polynomial Chaos Multi-Body Dynamics (PCMBoD) automation processes. To show additional functionality with PCMBoD, three additional examples are solved: a planar motorcycle traversing a bump, a planar ball bouncing and a spherical mechanism, the Agile Eye [9]. These examples also illustrate some of the limitations of the PCT approach. Each example is compared to a Monte Carlo simulation to verify its accuracy. The slider-crank also is solved numerically along with the additional example problems. The example problems try to use typical functionality in a multi-body dynamics (MBD) formulation.

The examples are simulated on a computer with the following specifications:

- Operating System – Windows 10,
- System Type – 64 bit,
- Processor - Intel Core i5-3330 CPU @3.00Ghz, and
- RAM – 8.00 GB.

The software used is:

- Matlab 2016a,
- Maple 2016, and
- Sundials 2.4.0 .

5.1 Slider-Crank Numerical Solution

The slider-crank example from Sections 3.7 and 4.3 is used to compare PCT to traditional methods of the Monte Carlo (MC) analysis. The deterministic slider-crank equations of motion are derived in Section 3.7, and the PCT equations of motion are derived in Section 4.3. Both of these formulations are derived using the automation processes DMBoD and PCMBoD, respectively. Using the deterministic slider-crank model, an MC analysis is performed and compared to the results of the PCT system.

5.1.1 Slider-Crank Problem Statement

The full problem statement is restated with numerical values for completeness. The slider-crank is shown in Fig. 3.3. Variability in the link lengths is incorporated by simulating populations of each with normal distributions: means of $\bar{l_1} = 1$ and $\bar{l_2} = 1.5$ and standard deviations of $\sigma_{l1} = 0.05$ and $\sigma_{l2} = 0.075$, respectively. Points *A* and *B* are revolute joints, and Point *C* is constrained to move in a slot. The mechanism moves in a plane perpendicular to the gravity vector; thus, gravitational forces are not important. The

mass of link 1 and link 2 are $m_1 = 2$ and $m_2 = 3$, respectively, and the moment of inertia is assumed to be a slender rod (i.e., $J_i = \frac{1}{3} m_i \bar{l_i}^2$). The mass and the moment of inertia are assumed to be constant as the link lengths vary. The location of the center of mass does not change because it is always the midpoint of the links. The system is driven by a torque at Point *A*,

$$T_{app} = \begin{cases} 1 & t \leq 1 \\ -1 & 1 < t \leq 2 \\ 0 & t > 2 \end{cases} . \tag{5.1}$$

The initial conditions are $\theta_{1o} \equiv \frac{\pi}{4}$ and $\dot{\theta}_{1o} \equiv 0$.

A DAE integrator is very sensitive to discontinuities. The torque as it is defined as a step function in Eqn. (5.1) needs to be represented by a smooth approximation. Following the approach in MSC.Adams, the step function is approximated as [37]

$$\mathrm{step}\left(t, t_1, x_1, t_2, x_2\right) = \begin{cases} x_1 & t \leq t_1 \\ x_2 + \left(x_2 - x_1\right)\left(3 - 2\Delta\right)\Delta^2 & t_1 < t < t_2 \\ x_2 & t \geq t_2 \end{cases} , \tag{5.2}$$

where $\Delta = \left(\frac{t - t_1}{t_2 - t_1}\right)$.

5.1.2 Slider-Crank Monte Carlo

In Section 3.7, the Matlab functions needed to solve the deterministic slider-crank are exported. Matlab is used to simulate the system. The first step is to enter the numerical values for the slider-crank problem. The next step finds a consistent set of initial conditions. Finally, the system is simulated, and the responses can be plotted.

Performing an MC analysis with the DMBoD requires some extra steps. In Section 3.7, the deterministic slider-crank equations of motion are derived. A database of 500 cases is built using the *randn* MATLAB function with different combinations for l_1 and l_2. A larger set of 1000 cases is built with similar results as the 500 case MC analysis. The resulting histogram for l_1 and l_2 can be seen in Figs. 5.1 and 5.2, respectively. Each case is simulated using the deterministic state space equations in Eqn. (3.39) with the applied torque in Eqn. (5.1) using the torque approximation in Eqn. (5.2).

5.1.3 Polynomial Chaos Theory Slider-Crank

In Section 4.3, the Matlab functions needed to solve the PCT slider-crank are exported. The PCT system derived is simulated using the state space equations in Eqn. (3.39). Additional initial conditions are needed, as the PCT slider crank has three DOF. The additional initial conditions for the slider-crank are the variations on θ_1 and $\dot{\theta}_1$. These additional initial conditions are zero.

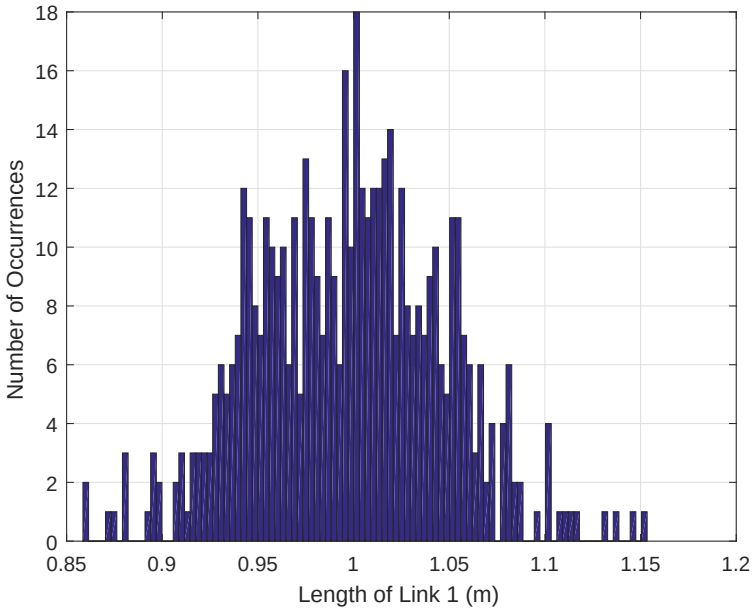

Figure 5.1: Slider-crank link 1 500 sample histogram for MC analysis

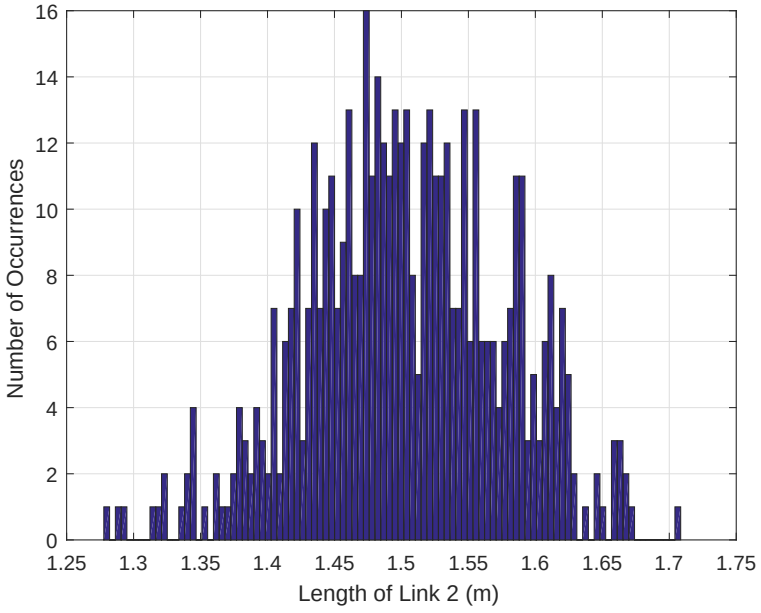

Figure 5.2: Slider-crank link 2 500 sample histogram for MC analysis

Matlab is used to simulate the system. The first step, as in the deterministic case, is to enter the numerical values for the slider-crank problem. The next step is to find a consistent set of initial conditions using the stated and additional PCT initial conditions. Finally, the system is simulated, and the responses can be plotted.

5.1.4 Slider-Crank Results and Comparison

The position responses for the MC and PCT analysis can be seen in Figs. 5.3 and 5.4. Instead of plotting the planar quaternions, the planar rotation angle is computed from the quaternion states. The position comparison between the two methods shows good agreement. The maximum L2-norm of the difference between the MC and the PCT responses for the translational displacement mean and standard deviations are 6.77×10^{-2} m and 7.01×10^{-2} m, respectively. The maximum L2-norm of the difference between the MC and the PCT responses for the angular displacement mean and standard deviations are 3.08×10^{-2} rad and 2.25×10^{-2} rad, respectively. In most instances, the blue PCT results lie on top of the red MC results, showing agreement between the two methods.

5.1.5 Uncertainty Effects

Depending on the outputs of the model, the uncertainty in the link lengths has different effects. As an example, the operational envelope of the mechanism is used in Fig. 3.3. To be more specific, the vertical displacement of Point B and the horizontal

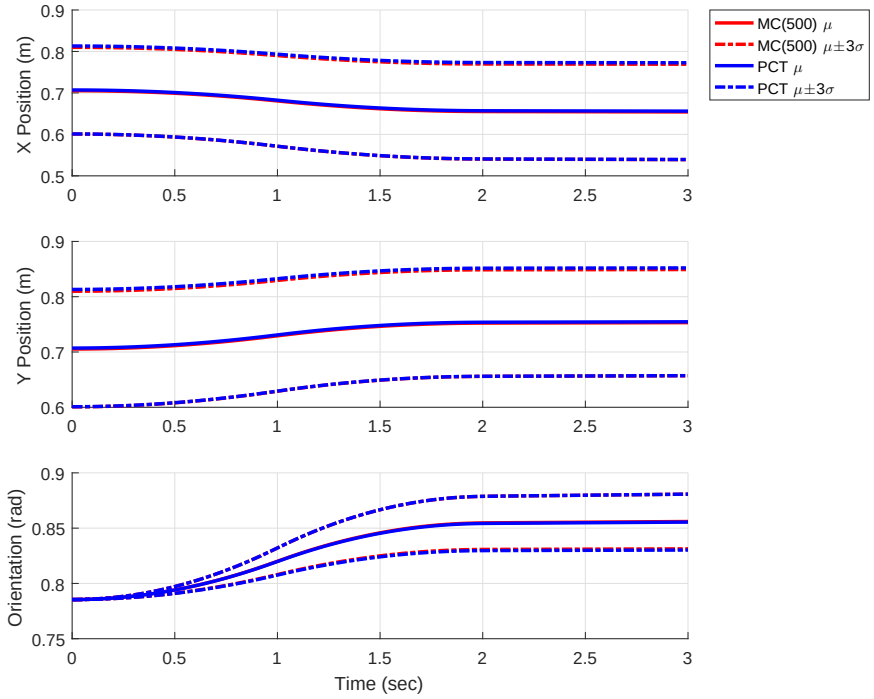

Figure 5.3: Slider-crank link 1 position MC and PCT comparison. The red lines are the MC analysis, and the blue lines are the PCT analysis. The orientation starts out with zero standard deviation because it is specified as an initial condition

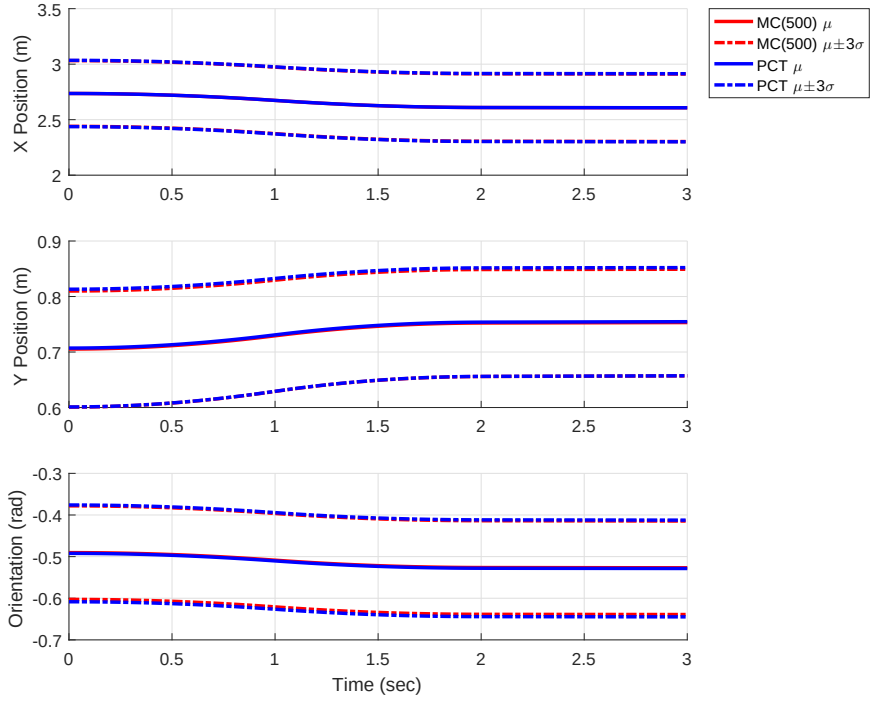

Figure 5.4: Slider-crank link 2 position MC and PCT comparison. The red lines are the MC analysis, and the blue lines are the PCT analysis.

displacement of Point C are used. Fig. 5.5 shows the standard deviation as a function of time of the vertical position Point B. The overall variation does not change because as the standard deviation of Link 1 decreases, it increases for Link 2. Looking at the horizontal standard deviation of Point C in Fig. 5.6, it can be seen that the overall standard deviation is driven by Link 1; Link 2 has no effect.

The slider-crank example shows the power of PCT when compared to the traditional Monte Carlo analysis. For this example, the equations of motion take about the same time to derive. The PCT slider-crank takes negligibly longer to solve than one Monte Carlo simulation; the complete Monte Carlo analysis takes 500 times longer. The same responses from the states are determined using both processes, with PCT being much quicker. In the next example, the process is applied to a real-world motorcycle simulated as a planar problem. This example increases the number of bodies in the problem and type of forces that appear in the problem.

5.2 Motorcycle Traversing a Bump

The complexity of the problem is increased with a motorcycle traversing a bump. The problem is still two-dimensional (2D); however, the number of bodies is increased to five. Also, there is potential energy, both gravitational and elastic potential. This example tests the integrator with the stiffness of the system increasing and tests the flexibility of PCMBoD.

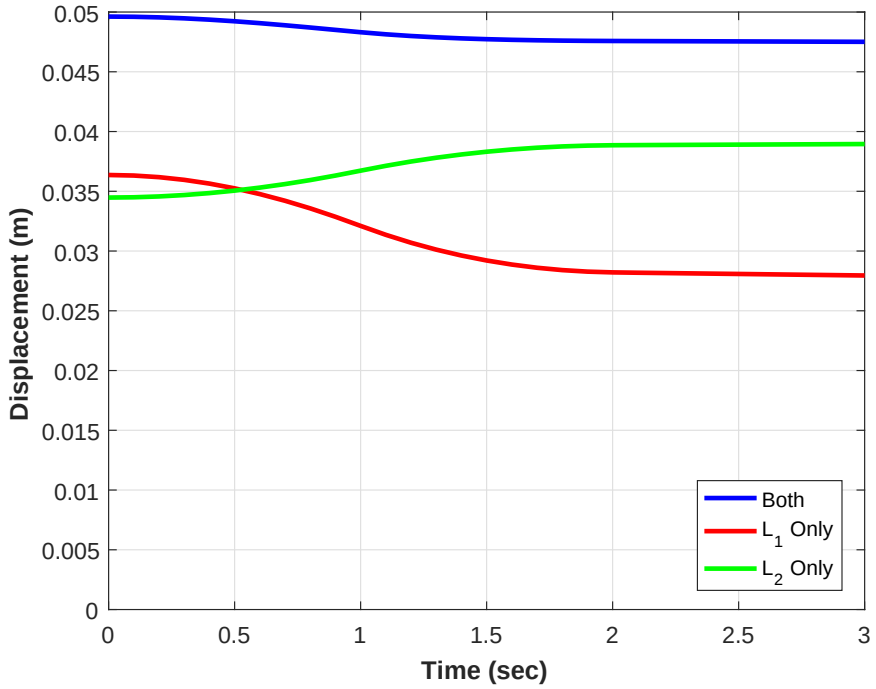

Figure 5.5: Slider-crank point B vertical displacement standard deviation quantified by using PCT. Each link length effects the overall standard deviation, which stays constant

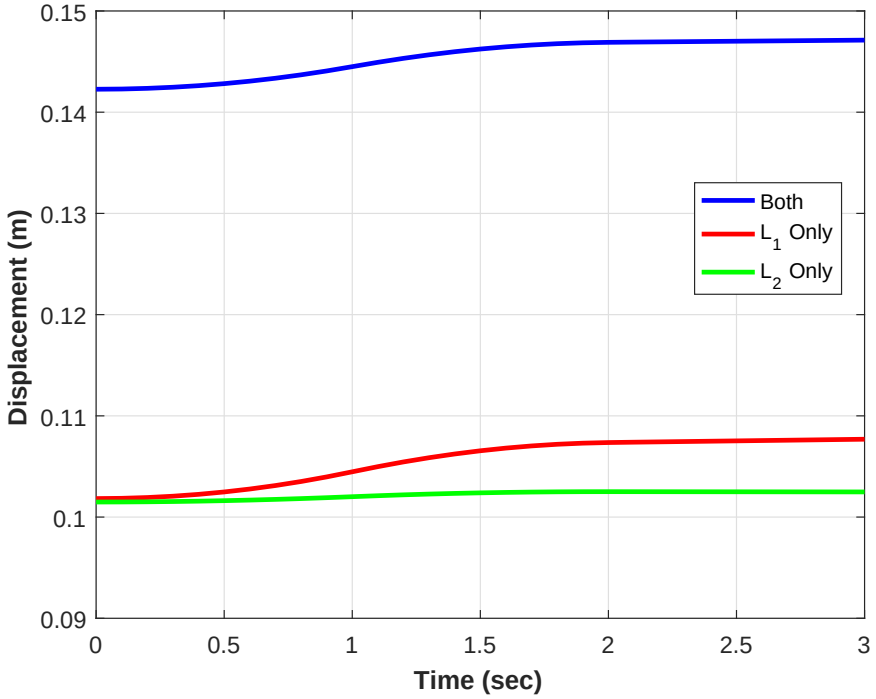

Figure 5.6: Slider-crank point C horizontal standard deviation quantified by using PCT. Link 1 only affects the standard deviation of the overall standard deviation

The motorcycle model is based on the work of Sharp [50] and is shown in Fig. 5.7. The numerical parameters for the motorcycle are found in Appendix C. The motorcycle is modeled as a planar system with rotations. The uncertainty in the system is from variation in the spring stiffness of the front fork and the rear shock. The motorcycle travels over a bump, which is modeled as moving points p_6 and p_7. The responses for the system that are important are the vertical and rotational displacement of the main body, 1.

The position initial conditions for the system are in the configuration shown with zero velocity.

The bump is created using Eqn. (5.2) for both the front and rear tires. Specifically,

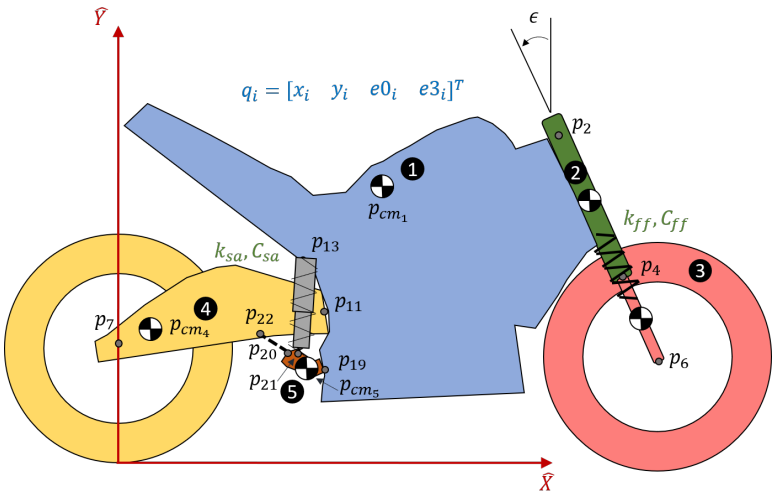

Figure 5.7: Motorcycle problem description

the bumps are modeled as

$$\begin{aligned} y_{\text{front}} = &\text{step}\left(t, t_{\text{start}}, 0, t_{\text{start}} + \frac{L_{\text{bump}}}{2v_{\text{odom}}}, h_{\text{bump}}\right) \\ &- \text{step}\left(t, t_{\text{start}} + \frac{L_{\text{bump}}}{2v_{\text{odom}}}, 0, t_{\text{start}} + \frac{L_{\text{bump}}}{v_{\text{odom}}}, h_{\text{bump}}\right) \text{ , and} \end{aligned} \tag{5.3}$$

$$\begin{aligned} y_{\text{rear}} = &\text{step}\left(t, t_{\text{start}}, 0, t_{\text{start}} + \frac{L_{\text{bump}}}{2v_{\text{odom}}} + \frac{L_{wheel}}{v_{\text{odom}}}, h_{\text{bump}}\right) \\ &- \text{step}\left(t, t_{\text{start}} + 2\frac{L_{\text{bump}}}{v_{\text{odom}}}, 0, t_{\text{start}} + \frac{L_{\text{bump}}}{v_{\text{odom}}} + \frac{L_{wheel}}{v_{\text{odom}}}, h_{\text{bump}}\right) , \end{aligned} \tag{5.4}$$

where t_{start} is the simulation time at which the front tire hits the bump, L_{bump} is the length of the bump, h_{bump} is the height of the bump, and v_{odom} is the speed of the motorcycle. The resulting displacement of each point can be seen in Fig. 5.8.

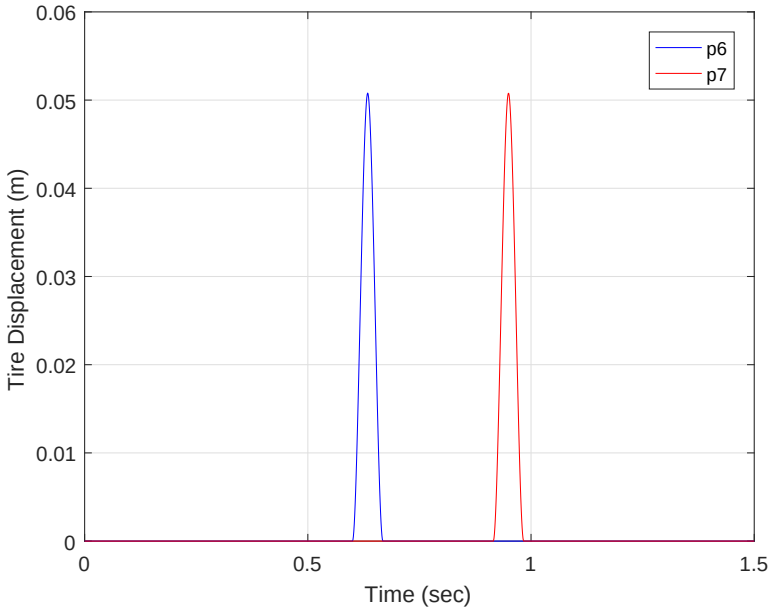

Figure 5.8: Motorcycle going over a bump at 4.4704 m/sec. The blue line is displacement of the front tire via p6 and the red line is the displacement of the rear tire via p7.

5.2.1 Inputs for the Motorcycle

The inputs to the system are

- $\underline{g} = \begin{bmatrix} 0 & -1 \end{bmatrix}^T$,
- $n_{\text{dof}} = 4$,
- $n_{\text{bod}} = 5$,
- $n_{\text{con}} = 13$, and
- $n_{\text{pct}} = 2$.

The constraints for the system are summarized in Table 5.1. **I** is the identity matrix, L_{2022} is the distance between p_{20} and p_{22}, and $R_{2D}(\epsilon)$ is a 2D rotation an angle of ϵ. Recall, $\underline{q}_{Ti}$ are the translational generalized coordinates for body *i*. In addition to the constraints, the forces need to be entered. There are four forces, two springs and two dampers, from the two shocks in the system. The first step to enter these forces is to describe the position and velocity of each point. For the front shock, the positions of the end points are

$$r_{\text{frontA}} = \underline{q}_{T2} + A_2 \left(\underline{p}_4 - \underline{p}_{cm_2} \right) \text{, and} \tag{5.5}$$

$$r_{\text{frontB}} = \underline{q}_{T3} + A_3 \left(\underline{p}_4 - \underline{p}_{cm_3} \right) , \tag{5.6}$$

Table 5.1: Motorcycle kinematic constraints

Type	Body *i*	Body *j*	Definition
Fix	2	1	$\Phi_{\text{fix}}\left(\underline{q}_{T2}, \mathbf{A}_2, \underline{p}_2 - \underline{p}_{cm_2}, \underline{q}_{T1}, \mathbf{A}_1, \underline{p}_2 - \underline{p}_{cm_1}\right)$
Revolute	4	1	$\Phi_{\text{rev}}\left(\underline{q}_{T4}, \mathbf{A}_4, \underline{p}_{11} - \underline{p}_{cm_4}, \underline{q}_{T1}, \mathbf{A}_1, \underline{p}_{11} - \underline{p}_{cm_1}\right)$
Revolute	5	1	$\Phi_{\text{rev}}\left(\underline{q}_{T5}, \mathbf{A}_5, \underline{p}_{19} - \underline{p}_{cm_5}, \underline{q}_{T1}, \mathbf{A}_1, \underline{p}_{19} - \underline{p}_{cm_1}\right)$
Fxd Dist	5	4	$\Phi_{\text{fd}}\left(\underline{q}_{T5}, \mathbf{A}_5, \underline{p}_{20} - \underline{p}_{cm_5}, \underline{q}_{T4}, \mathbf{A}_4, \underline{p}_{22} - \underline{p}_{cm_4}, L_{2022}\right)$
Prismatic	3	2	$\Phi_{\text{pris}}\left(\underline{q}_{T3}, \mathbf{A}_3, \underline{p}_4 - \underline{p}_{cm_3}, R_{2D}\left(\epsilon\right), \underline{q}_{T2}, \mathbf{A}_2, \underline{p}_4 - \underline{p}_{cm_2}, R\left(\epsilon\right)\right)$
Dot-2	1	Grnd	$\Phi_{d2}\left(\underline{q}_{T1}, \mathbf{A}_1, \underline{0}, \underline{0}, \mathbf{I}, \underline{0}, \mathbf{I}\right)$
Driver	1	Grnd	$\Phi_{d2}\left(\underline{q}_{T3}, \mathbf{A}_3, \underline{p}_6 - \underline{p}_{cm_3}, \underline{0}, \mathbf{I}, \underline{0}, R\left(\frac{\pi}{2}\right)\right) - p_{6_y} - y_{\text{front}}$
Driver	1	Grnd	$\Phi_{d2}\left(\underline{q}_{T4}, \mathbf{A}_4, \underline{p}_7 - \underline{p}_{cm_4}, \underline{0}, \mathbf{I}, \underline{0}, R\left(\frac{\pi}{2}\right)\right) - p_{7_y} - y_{\text{rear}}$

and the velocities of the end points are

$$\dot{r}_{\text{frontA}} = \dot{\underline{q}}_{T2} + A_2\left(\left(2\mathbf{G}_2\dot{\underline{e}}_2\tilde{\mathbf{\Omega}}\right)\left(\underline{p}_4 - \underline{p}_{cm_2}\right)\right) \text{ , and} \tag{5.7}$$

$$\dot{r}_{\text{frontB}} = \dot{\underline{q}}_{T3} + A_3\left(\left(2\mathbf{G}_3\dot{\underline{e}}_3\tilde{\mathbf{\Omega}}\right)\left(\underline{p}_4 - \underline{p}_{cm_3}\right)\right) , \tag{5.8}$$

where

$$\tilde{\mathbf{\Omega}} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} . \tag{5.9}$$

For the rear shock, the positions of the end points are

$$r_{\text{rearA}} = \underline{q}_{T1} + A_1\left(\underline{p}_{13} - \underline{p}_{cm_1}\right) \text{ , and} \tag{5.10}$$

$$r_{\text{rearB}} = \underline{q}_{T5} + A_5\left(\underline{p}_{21} - \underline{p}_{cm_5}\right) , \tag{5.11}$$

and the velocities are

$$\dot{r}_{\text{rearA}} = \dot{\underline{q}}_{T1} + A_1 \left(\left(2\mathbf{G}_1 \dot{\underline{e}}_1 \tilde{\mathbf{\Omega}} \right) \left(\underline{p}_{13} - \underline{p}_{cm_1} \right) \right) \text{ , and} \tag{5.12}$$

$$\dot{r}_{\text{rearB}} = \dot{\underline{q}}_{T5} + A_5 \left(\left(2\mathbf{G}_5 \dot{\underline{e}}_5 \tilde{\mathbf{\Omega}} \right) \left(\underline{p}_{21} - \underline{p}_{cm_5} \right) \right) \text{ .} \tag{5.13}$$

To define the forces, the differences in the positions and the velocities are needed,

$$\Delta r_{\text{front}} = r_{\text{frontB}} - r_{\text{frontA}} ma \tag{5.14}$$

$$\Delta r_{\text{rear}} = r_{\text{rearB}} - r_{\text{rearA}} \text{ ,} \tag{5.15}$$

$$\Delta \dot{r}_{\text{front}} = \dot{r}_{\text{frontB}} - \dot{r}_{\text{frontA}} \text{ , and} \tag{5.16}$$

$$\Delta \dot{r}_{\text{rear}} = \dot{r}_{\text{rearB}} - \dot{r}_{\text{rearA}} \text{ .} \tag{5.17}$$

For the damper of the shocks, the forces are

$$F_1 = -C_{ff} \Delta \dot{r}_{\text{front}} \text{ , and} \tag{5.18}$$

$$F_2 = -C_{sa} \Delta \dot{r}_{\text{rear}} \text{ .} \tag{5.19}$$

The corresponding velocities for the Principle of Virtual Work are

$$v_1 = \Delta \dot{r}_{\text{front}} \text{ , and} \tag{5.20}$$

$$v_2 = \Delta \dot{r}_{\text{rear}} \text{ .} \tag{5.21}$$

The last item that needs to be entered is the elastic potential energy. For the motorcycle, there are two shocks and therefore two springs. If L_{sa} is the free length of the rear spring, the elastic potential for the system is

$$V_{\text{elastic}} = \frac{1}{2}k_{ff}\left(\Delta r_{\text{front}}\right)^T\left(\Delta r_{\text{front}}\right) + \frac{1}{2}k_{sa}\left(\left(\left(\Delta r_{\text{rear}}\right)^T\left(\Delta r_{\text{rear}}\right)\right)^{\frac{1}{2}} - L_{sa}\right)^2 . \quad (5.22)$$

Next, distributions and corresponding parameters are provided for the random variables and chosen orthogonal polynomial basis as shown in Table 5.2.

Table 5.2: Motorcycle random variables and PCT parameters

Variable	Distribution	Mean (μ)	Std Dev σ	Orthogonal Polynomial Basis
k_{ff}	Normal	$\bar{k_{ff}}$	$\sigma_{k_{ff}}$	Hermite
k_{sa}	Normal	$\bar{k_{sa}}$	$\sigma_{k_{sa}}$	Hermite

Finally, the last input to be entered is the definition of the outputs of the system,

$$\text{output}_1 = y_1 \text{ , and} \quad (5.23)$$

$$\text{output}_2 = \text{atan2}\left(2\, e0_i{}^2 - 1, 2\, e0_i\, e3_i\right) . \quad (5.24)$$

5.2.2 Computational Limitations Solving the Motorcycle Traversing a Bump

With the inputs of the system fully defined, DMBoD can be used to solve the problem; however, PCMBoD cannot. The elastic potential in Eqn. (5.22) cannot be

projected onto the orthogonal polynomial basis. When using Variational Work Eqn. (4.6) with Eqn. (5.22), the resulting energy from the calculation is no longer real; it is complex. The complex number comes from projecting the square root. The Galerkin projection method described in this dissertation does not work explicitly with rational functions of polynomials, square roots or transcendental functions [10]. The Galerkin projection in this dissertation can handle a regular polynomial. In this case, with the square root present, a different calculation can be performed to project it onto the orthogonal polynomial basis. To handle the projection of the square root of the form [10]

$$Y\left(\underline{\xi}\right) = X\left(\underline{\xi}\right)^{\frac{1}{2}} \ , \tag{5.25}$$

a iterative algorithm using Newton-Raphson is used. To solve Eqn. (5.25), it is rewritten [10]

$$X\left(\underline{\xi}\right) = Y\left(\underline{\xi}\right)^{2} \ , \tag{5.26}$$

and the coefficients of $Y\left(\underline{\xi}\right)$ are estimated. Special cases for the Galerkin projection are outside the scope of this dissertation.

To solve this example, Eqn. (5.22) is modified to assume that the rear shock free length is zero. This yields an acceptable rational polynomial

$$V_{\text{elastic}} = \frac{1}{2} k_{ff} \left(\Delta r_{\text{front}}\right)^{T} \left(\Delta r_{\text{front}}\right) + \frac{1}{2} \left(\Delta r_{\text{rear}}\right)^{T} \left(\Delta r_{\text{rear}}\right) \ . \tag{5.27}$$

The MC analysis is performed on both systems, one with Eqn. (5.22) and one with Eqn. (5.27). The PCT method is only performed on the system with Eqn. (5.27).

Another issue in simulating the motorcycle is choosing a bump model that can be simulated. As the speed is increased or the length or height of the bump changes, the integrator *IDAS* from *sundialsTB* fails. Regardless of how much the integrator tolerances are adjusted, both looser and tighter, the variable step integrator goes to a very small time step, less than 1×10^{-13} sec. The integrator only completes the simulation successfully if the bump is relatively small, and the velocity of the motorcycle is slow. This restriction is due to increased stiffness of the numerical system, where a small change in the input caused a large output in the states of the model. One way to address the stiffness is to specify the Jacobian for the system.

Using the current settings, the Jacobian for the system is estimated numerically. Given the general arbitrary system

$$\underline{f}\left(\dot{\underline{x}}, \underline{x}, t\right) = \underline{0}\,, \tag{5.28}$$

the Jacobian can be computed by [23]

$$\mathbf{J}_{\text{sys}} = \alpha \frac{\partial \underline{f}}{\partial \dot{\underline{x}}} + \frac{\partial \underline{f}}{\partial \underline{x}}\,, \tag{5.29}$$

where α is a scalar that is a function of the current simulation step size. Using the residual

in Eqn. (3.40), the system Jacobian is

$$\mathbf{J}_{\text{sys}} = \alpha \mathbf{J}_v + \mathbf{J}_p \ , \tag{5.30}$$

where

$$\mathbf{J}_v = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{M} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \ , \text{ and} \tag{5.31}$$

$$\mathbf{J}_p = \begin{bmatrix} \frac{\partial\left(\mathbf{\Phi}_{\boldsymbol{q}}^T \underline{x}_\mu\right)}{\partial \underline{x}_p} & \mathbf{I} & \mathbf{0} & \mathbf{\Phi}_{\boldsymbol{q}}^T \\ \frac{\partial\left(-\mathbf{M}\dot{\underline{x}}_v + \underline{Q} - \underline{C} + \mathbf{\Phi}_{\boldsymbol{q}}^T \underline{x}_\lambda\right)}{\partial \underline{x}_p} & \frac{\partial\left(\underline{Q} - \underline{C}\right)}{\partial \underline{x}_v} & \mathbf{\Phi}_{\boldsymbol{q}}^T & \mathbf{0} \\ \mathbf{\Phi}_{\boldsymbol{q}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \frac{\partial(\mathbf{\Phi}_{\boldsymbol{q}} \underline{x}_v + \underline{\Phi}_t)}{\partial \underline{x}_p} & \mathbf{\Phi}_{\boldsymbol{q}} & \mathbf{0} & \mathbf{0} \end{bmatrix} . \tag{5.32}$$

Unfortunately, although most of the components in the system Jacobian are already computed, the terms not computed are not trivial to calculate. The time Maple took to create these matrices and export them as Matlab functions is significantly longer than not exporting them. Computing and exporting these values takes much longer than running the complete MC analysis. Since this process eliminated the cycle time benefits when comparing to the MC analysis, the effort is abandoned as a solution to address the system stiffness.

5.2.3 Motorcycle Results Comparison and Discussion

To compare the outputs of the model between an MC analysis and the PCT analysis, 500 cases are used. For the MC analysis, the 500 cases apply to k_{ff} and k_{sa}. For the PCT analysis, the 500 cases are applied to ξ_1 and ξ_2. Recall that ξ_1 and ξ_2 are the random variables of the orthogonal polynomial basis which resulted from projecting k_{ff} and k_{sa}, respectively. The comparison between responses of the two analyses can be seen in Figs. 5.9 and 5.10. For the zero length rear spring case, the PCT lies directly on top of the MC analysis, making the MC analysis almost invisible. The L2-norm of the difference between the MC and the PCT responses for the main body vertical displacement mean and standard deviations are 2.83×10^{-5} m and 2.90×10^{-5} m, respectively. The L2-norm of the difference between the MC and the PCT responses for the main body angular displacement mean and standard deviations are 1.09×10^{-4} rad and 5.84×10^{-5} rad, respectively. The green lines for the zero length MC analysis results appear to not be plotted. Unfortunately, both of these analyses are not as accurate as the original MC analysis using a non-zero length for the rear shock. Systems containing a square root cannot be handled by PCMBoD; special Galerkin projection cases are not yet implemented. The square root in the original elastic potential in Eqn. (5.22) is not able to be projected. Hence, an inaccurate simplification is made. In the end, for this example, PCT is quicker to set up and faster to run than the MC analyses, approximately ten times faster; however, accuracy is sacrificed.

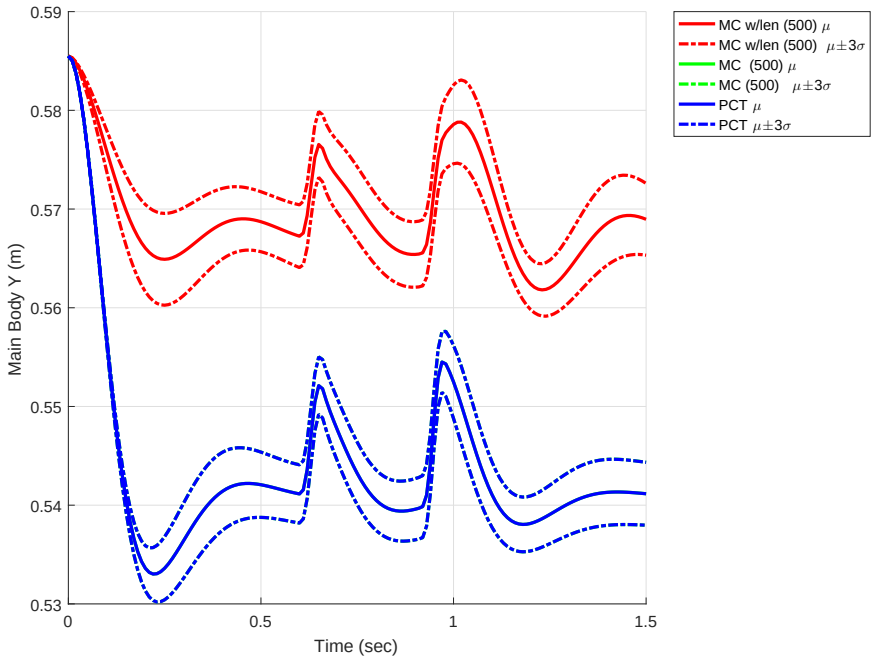

Figure 5.9: Motorcycle body 1 vertical response MC and PCT comparison. The red lines are the MC analysis of the non zero length spring, the green lines are the MC analysis of a zero length spring, and the blue lines are the results of the PCT analysis.

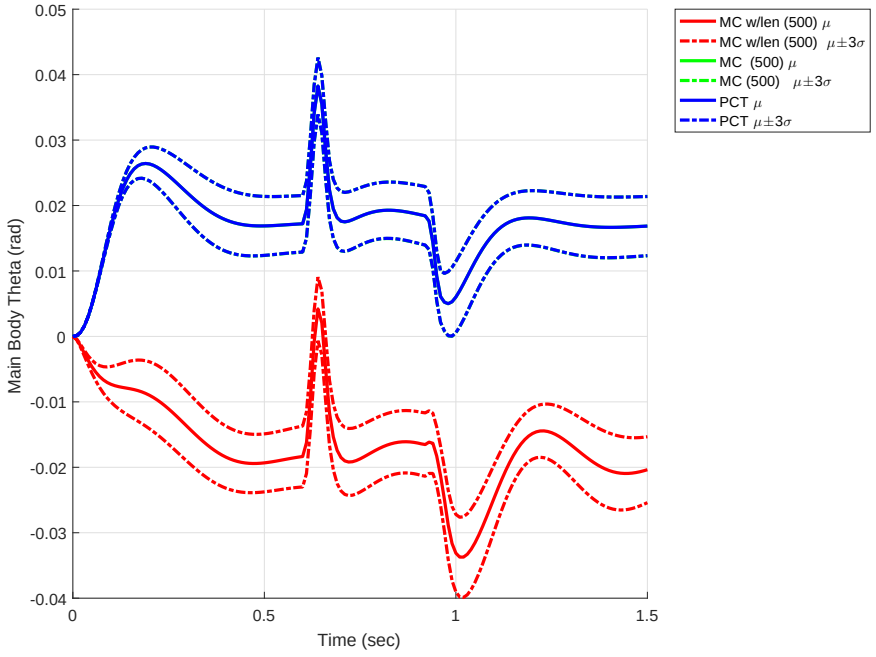

Figure 5.10: Motorcycle body 1 rotational response MC and PCT comparison. The red lines are the MC analysis of the non zero length spring, the green lines are the MC analysis of a zero length spring, and the blue lines are the results of the PCT analysis.

Increasing the complexity of the problem proved to be an issue for the PCT formulation and the integrator in the motorcycle example. A better integrator allows the system to traverse a more severe bump. Without a change to the PCMBoD automation process to handle springs of non-zero lengths, the utility of PCMBoD is limited in these cases with the Galerkin projection.

In the next example, the DCMBoD and PCMBoD processes are applied to a system with contact. It is a simple example, but it further shows the capability of the PCT method and its limitations.

5.3 Bouncing Ball

A common type of force found in an MBD system is a discontinuous force, such as contact or friction. A bouncing ball is used as example to show how a system with contact and uncertainty is analyzed. A ball is dropped from a known distance and allowed to bounce with a perfect elastic contact. The bouncing ball is shown in Fig. 5.11. The uncertainty in the system is from variation in the radius, r, and contact stiffness, k, and is incorporated by simulating populations of each with normal distributions: means of $\bar{r} = 1$ and $\bar{k} = 1 \times 10^6$ and standard deviations of $\sigma_r = 0.05$ and $\sigma_k = 3.33 \times 10^3$, respectively. The ball is constrained to move only vertically. Gravity acts in the negative $\widehat{Y}$ direction.

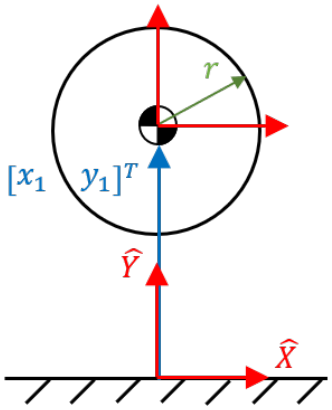

Figure 5.11: Bouncing ball problem description

The mass of ball is $m = 1$. The contact force is modeled as a linear spring

$$F_{\text{contact}} = \begin{cases} 0 & y_1 - r \geq 0 \\ -k\left(y_1 - r\right) & y_1 - r < 0 \end{cases}. \tag{5.33}$$

5.3.1 Inputs for the Bouncing Ball

The inputs to the system are

- $\underline{g} = \begin{bmatrix} 0 & -1 \end{bmatrix}^T$,
- $n_{\text{dof}} = 2$,
- $n_{\text{bod}} = 1$,
- $n_{\text{con}} = 1$, and

- $n_{\text{pct}} = 2$.

The constraint for the system is a dot-2 constraint,

$$\Phi = \Phi_{d2}\left(\underline{r}_1, \mathbf{I}, \underline{0}, \underline{0}, \mathbf{I}, \underline{0}, R_{2D}\left(0\right)\right) \ . \tag{5.34}$$

There is only one force in the system, the contact force. For the automation process, Eqn. (5.33) becomes

$$F_1 = \begin{cases} 0 & y_1 - r \geq 0 \\ \\ -k\left(y_1 - r\right) & y_1 - r < 0 \end{cases} . \tag{5.35}$$

The velocity of the point of contact is

$$v_1 = \dot{y}_1 \ . \tag{5.36}$$

Next, distributions and corresponding parameters are provided for the random variables and chosen orthogonal polynomial basis as shown in Table 5.3. Finally, the last

Table 5.3: Bouncing ball random variables and PCT parameters

Variable	Distribution	Mean (μ)	Std Dev σ	Orthogonal Polynomial Basis
r	Normal	$\bar{r}$	σ_r	Hermite
k	Normal	$\bar{k}$	σ_k	Hermite

input to be entered is the definition of the outputs of the system. The output is defined as

$$\text{output}_1 = y_1 \ . \tag{5.37}$$

5.3.2 Computational Limitations Solving the Bouncing Ball

Because the force is piecewise, the function cannot be projected onto the orthogonal polynomial basis using Eqn. (4.4). In [60], the PCT Hermite polynomials are described as the continuous polynomial chaos. An additional constraint is that PCT needs to be a globally smooth function [61]. A contact force is discontinuous the way it is modeled typically in an MBD system. In this case, the contact is continuous, but not smooth. Unfortunately, Eqn. (5.35) cannot be modeled in this form and be consistent with the current application of PCT. Another process needs to be applied for piecewise functions, such as the piecewise polynomial method described by [2]. As an assumption, to project the piecewise force conditions, only the means of the conditions are used.

Ultimately, when projected, the contact force in Eqn. (5.35) becomes

$$\mathring{F}_1 = \begin{bmatrix} \begin{cases} \langle 0, 1 \rangle & \langle y_1 - r, 1 \rangle \geq 0 \\ \langle -k\left(y_1 - r\right), 1 \rangle & \langle y_1 - r, 1 \rangle < 0 \end{cases} \\ \begin{cases} \langle 0, \xi_1 \rangle & \langle y_1 - r, 1 \rangle \geq 0 \\ \langle -k\left(y_1 - r\right), \xi_1 \rangle & \langle y_1 - r, 1 \rangle < 0 \end{cases} \\ \begin{cases} \langle 0, \xi_2 \rangle & \langle y_1 - r, 1 \rangle \geq 0 \\ \langle -k\left(y_1 - r\right), \xi_2 \rangle & \langle y_1 - r, 1 \rangle < 0 \end{cases} \end{bmatrix} . \tag{5.38}$$

The MC analysis is performed using Eqn. (5.35). The PCT method is performed with Eqn. (5.38).

5.3.3 Bouncing Ball Comparison and Results

To compare the outputs of the model between an MC analysis and the PCT analysis, 500 cases are used. For the MC analysis, the 500 cases apply to r and k. For the PCT analysis, the 500 cases are applied to ξ_1 and ξ_2. The comparison between these two analyses can be seen in Fig. 5.12. For the first contact the results comparison is very good; the PCT results lie directly on top of the MC results. With each subsequent contact, the PCT comparison diverges farther and farther from the MC. Focusing on the first two contacts, the L2-norm difference between the MC and PCT responses for the mean and

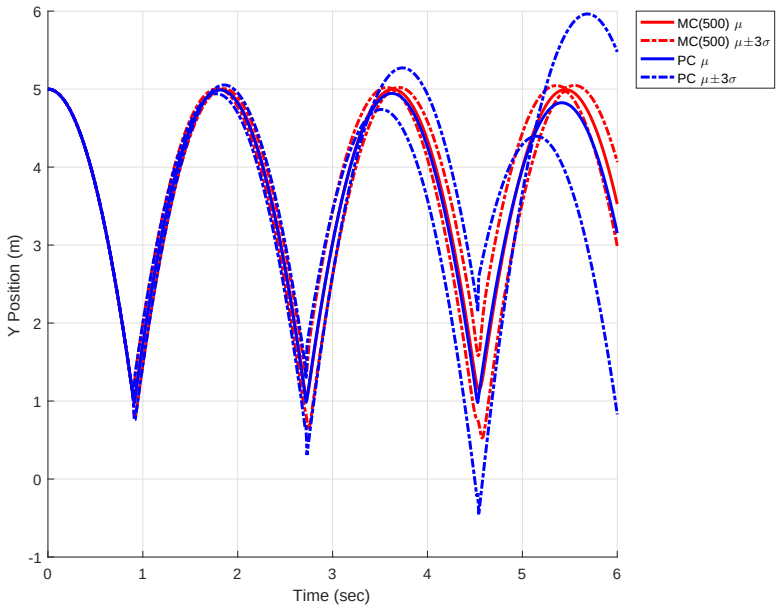

Figure 5.12: Bouncing ball body 1 vertical response MC and PCT comparison. The MC and PCT analysis agree in the beginning with PCT diverging from the MC analysis as time progresses.

standard deviation are 2.04×10^{-1} m and 3.48×10^{-3} m, respectively. Unfortunately, the overall L2-norm for the mean and standard deviation are 2.43 m and 4.00 m, respectively. Using the contact force approximation, the PCT solution seems to add energy to the system by rising above the starting point. The ball also penetrates the ground more and more when comparing to the MC analysis. The system is unstable, with each successive contact growing in magnitude. Adding a conservation of energy constraint to the system might help, but this is outside the dissertation scope. These unstable results are a direct result of the simplification of the piecewise force. A better method for projecting piecewise functions is needed.

In the next example, the process is be applied to a three-dimensional (3D) system.

This example increases the complexity of the system by adding more DOF to each body and moving through a 3D space.

5.4 Agile Eye

The complexity of the problem is increased again with the Agile Eye. The example is modeling a spherical mechanism in 3D. The Agile Eye example increases the dimensionality of the system and adds 3D rotations. This example is the final test in this dissertation for the flexibility of PCMBoD.

The Agile Eye example is based on the work of Caron [5] and is shown in Fig. 5.13. The Agile Eye mechanism is used to orient a camera. The numerical parameters for the Agile Eye are found in Appendix D. The Agile Eye is modeled in full 3D space. The uncertainty in the system is from variation in the location of the center of mass x and y coordinates of the camera, body 1 (see Fig. 5.13). The system is driven by a moment in each actuator. The response for the system that is important is the unit normal of the view of the camera. Gravity is ignored in this example. The initial conditions for the position are in the configuration shown with zero velocity.

The moment applied to body 2 is

$$\begin{aligned} M_A = &\operatorname{step}\left(t, 0.1 - \operatorname{tol}, 0, 0.1 + \operatorname{tol}, M_x\right) \\ &- \operatorname{step}\left(t, 0.2 - \operatorname{tol}, 0, 0.2 + \operatorname{tol}, M_x\right) \ , \end{aligned} \tag{5.39}$$

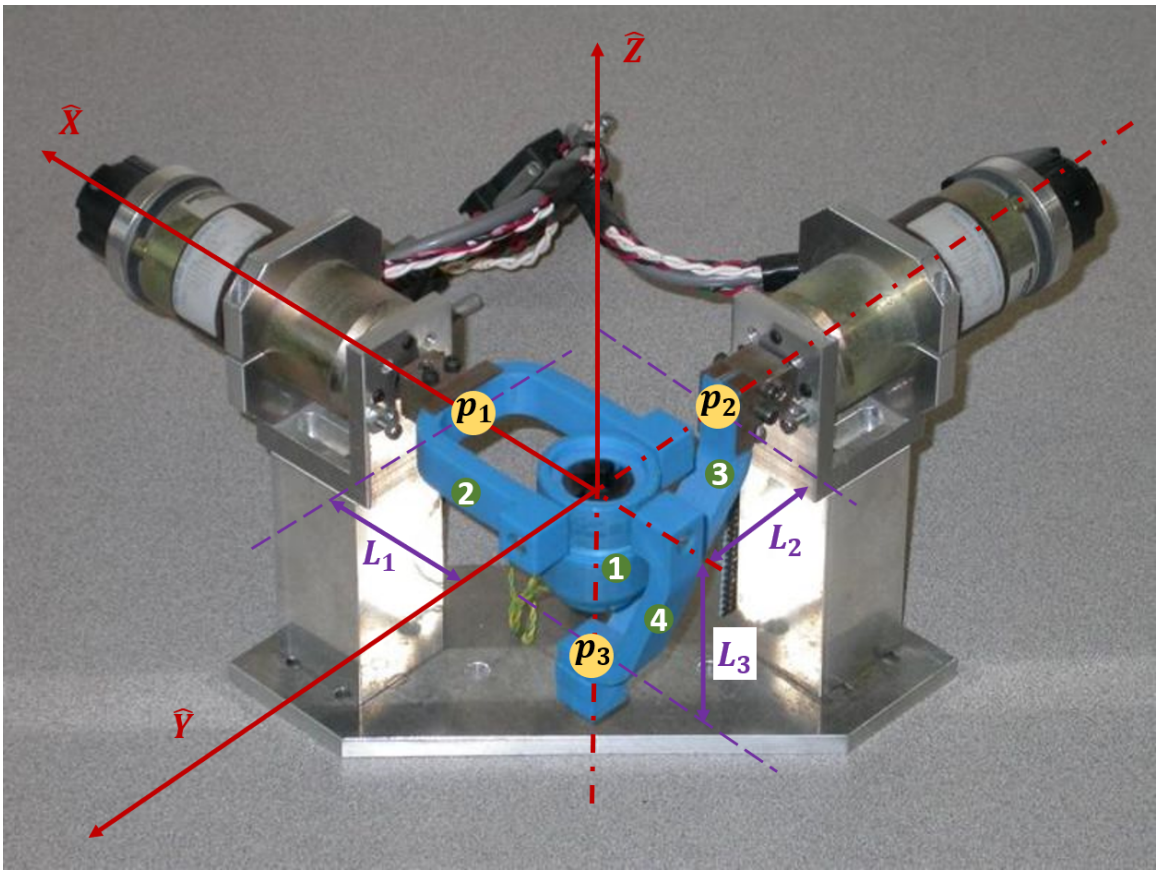

Figure 5.13: Agile Eye problem description [9]

and the moment applied to body 3 is

$$\begin{aligned} M_B = &\text{step}\left(t, 0.15 - \text{tol}, 0, 0.15 + \text{tol}, M_y\right) \\ &- \text{step}\left(t, 0.25 - \text{tol}, 0, 0.25 + \text{tol}, M_y\right) \ . \end{aligned} \tag{5.40}$$

5.4.1 Inputs for the Agile Eye

The inputs to the system are

- $\underline{g} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$,
- $n_{\text{dof}} = 7$,
- $n_{\text{bod}} = 4$,
- $n_{\text{con}} = 22$, and
- $n_{\text{pct}} = 2$.

The constraints for the system are summarized in Table 5.4. In addition to the constraints, the moments need to be entered. The first step to enter these moments is to describe the angular velocity of each body. For the actuators, the angular velocities are

$$\underline{\omega}_A = 2G_2\underline{e}_2 \text{ , and} \tag{5.41}$$

$$\underline{\omega}_B = 2G_3\underline{e}_3 \text{ .} \tag{5.42}$$

The next input is the elastic potential. For the Agile Eye, there is no elastic potential. Hence,

$$V_{\text{elastic}} = 0 \text{ .} \tag{5.43}$$

Next, distributions and corresponding parameters are provided for the random variables and chosen orthogonal polynomial basis as shown in Table 5.5. Finally, the last input to be entered is the definition of the output of the system, which is the direction the camera is

Table 5.4: Agile Eye kinematic constraints

Type	Body *i*	Body *j*	Definition
Revolute	2	Grnd	$\Phi_{\text{rev}}\left(\underline{q}_{T2}, \mathbf{A}_2, \underline{p}_1 - \underline{p}_{cm_2}, R\left(y, \frac{\pi}{2}\right), \underline{0}, \mathbf{I}, \underline{p}_1, R\left(y, \frac{\pi}{2}\right)\right)$
Revolute	3	Grnd	$\Phi_{\text{rev}}\left(\underline{q}_{T3}, \mathbf{A}_3, \underline{p}_2 - \underline{p}_{cm_3}, R\left(x, -\frac{\pi}{2}\right), \underline{0}, \mathbf{I}, \underline{p}_2, R\left(x, -\frac{\pi}{2}\right)\right)$
Revolute	1	2	$\Phi_{\text{rev}}\left(\underline{q}_{T1}, \mathbf{A}_1, -\underline{p}_{cm_1}, R\left(x, -\frac{\pi}{2}\right), \underline{q}_{T2}, \mathbf{A}_2, -\underline{p}_{cm_2}, R\left(x, -\frac{\pi}{2}\right)\right)$
Revolute	1	4	$\Phi_{\text{rev}}\left(\underline{q}_{T1}, \mathbf{A}_1, -\underline{p}_{cm_1}, R\left(y, \frac{\pi}{2}\right), \underline{q}_{T4}, \mathbf{A}_4, -\underline{p}_{cm_4}, R\left(y, \frac{\pi}{2}\right)\right)$
Parallel-2	4	3	$\Phi_{p2}\left(\underline{q}_{T3}, \mathbf{A}_3, \underline{p}_3 - \underline{p}_{cm_3}, \underline{q}_{T4}, \mathbf{A}_4, \underline{p}_3 - \underline{p}_{cm_4}, \mathbf{I}\right)$

Table 5.5: Agile Eye random variables and PCT parameters

Variable	Distribution	Mean (μ)	Std Dev σ	Orthogonal Polynomial Basis
x_{cm_1}	Normal	$\bar{x_{cm1}}$	$\sigma_{x_{cm1}}$	Hermite
y_{cm_1}	Normal	$\bar{y_{cm1}}$	$\sigma_{y_{cm1}}$	Hermite

pointed,

$$\text{output} = A_1 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} . \tag{5.44}$$

5.4.2 Computational Limitations Solving the Agile Eye

Finding a set of applied moments that allow the simulation to complete is very difficult for the PCT version of the Agile Eye. The deterministic model using DMBoD handles the moment successfully; there is always a solution. If the moments are increased by an order of magnitude, the PCMBoD formulated model fails at one of the smooth transitions of the applied moments (i.e., around $t = 0.1, 0.15, 0.2,$ or 0.25 sec). Like the

motorcycle example, regardless of how much the integrator tolerance is adjusted, the variable step integrator goes to a very small time step, less than 1×10^{-13} sec. To address this issue, the system Jacobian can be calculated explicitly using Eqn. (5.30) instead of relying on the numerically calculated Jacobian. Unfortunately, for the 3D case, the components in Eqn. (5.30) are even more computationally expensive when comparing to the motorcycle calculations. The computation and export takes hours to write out as Matlab functions, much longer than the MC analysis. Again, this negates any cycle time advantages that PCT has over the MC analysis. As a solution, the computation of the system Jacobian is abandoned.

Even though finding a set of applied moments that works with the Agile Eye, the system has to be moved using applied moments. If a kinematic driver is needed, this requires $\mathrm{atan2}$. The $\mathrm{atan2}$ function allows an angle to be computed on an interval of $-\pi \leq x \leq \pi$, where using $\sin$ limits the interval to $0 \leq x \leq \pi$, and $\cos$ to $-\frac{\pi}{2} \leq x \leq \frac{\pi}{2}$. The addition of a kinematic driver involves further refinement of the Taylor series expansion of a $\sin$ and $\cos$ function. It might not be possible to place a PCT Taylor Series expanded $\sin$ and $\cos$ into $\mathrm{atan2}$ and still be able to find an answer.

5.4.3 Agile Eye Results Comparison and Discussion

To compare the outputs of the model between an MC analysis and the PCT analysis, 500 cases are used. For the MC analysis, the 500 cases apply to x_{cm_1} and y_{cm_1}.

For the PCT analysis, the 500 cases are applied to ξ_1 and ξ_2. The comparison between these two analyses can be seen in Fig. 5.14. The comparison is not good due to the underlying distribution of the response. Although the means are identical, when looking at Fig. 5.14, the standard deviations do not compare. The maximum L2-norm of the difference between the MC and the PCT responses for the mean is 9.82×10^{-2} m, but for the standard deviation is 6.69×10^{-1} m. A time history histogram of the MC results is shown in Fig. 5.15. A time history histogram shows the histogram at each slice in time by using a color axis. When looking at Fig. 5.15, the response of the camera is not normally distributed. Unfortunately, a normally distributed response is an assumption for the

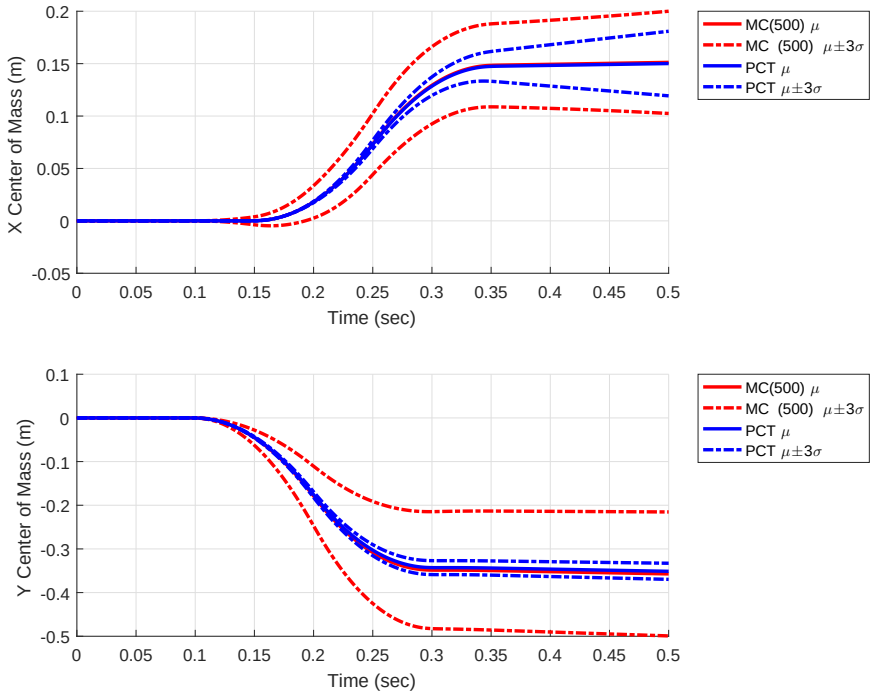

Figure 5.14: Agile Eye camera view response MC and PCT comparison. The red lines are the MC analysis and the blue lines are the PCT analysis. The means show good agreement, but the standard deviations do not.

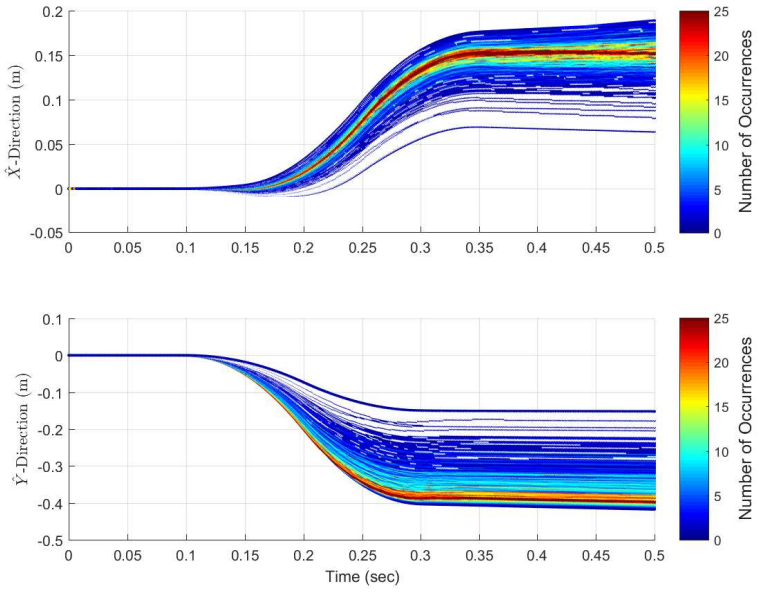

Figure 5.15: Agile Eye camera view MC time history histogram. The color shows the number of occurrences of at each time. The camera view from the MC analysis is not normally distributed.

solution by only expanding the states by first order Hermite polynomial expansions. A time slice of Fig. 5.15 at $t = 0.4$ sec can be seen in Fig. 5.16. This time slice illustrates even further that the response is not normally distributed. Using the FBD tool from the File Exchange of Matlab [1], the response appears to fit a Weibull distribution. The PCT time history histogram and a time slice at $t = 0.4$ sec histogram are shown in Fig. 5.17 and Fig. 5.18, respectively. These two plots clearly show that the PCT derived response are incorrectly normally distributed. The PCT response giving a normally distributed response is to be expected because of the first order Hermite polynomial expansion used.

Truncation of the orthogonal polynomial basis is always a concern with PCT. If

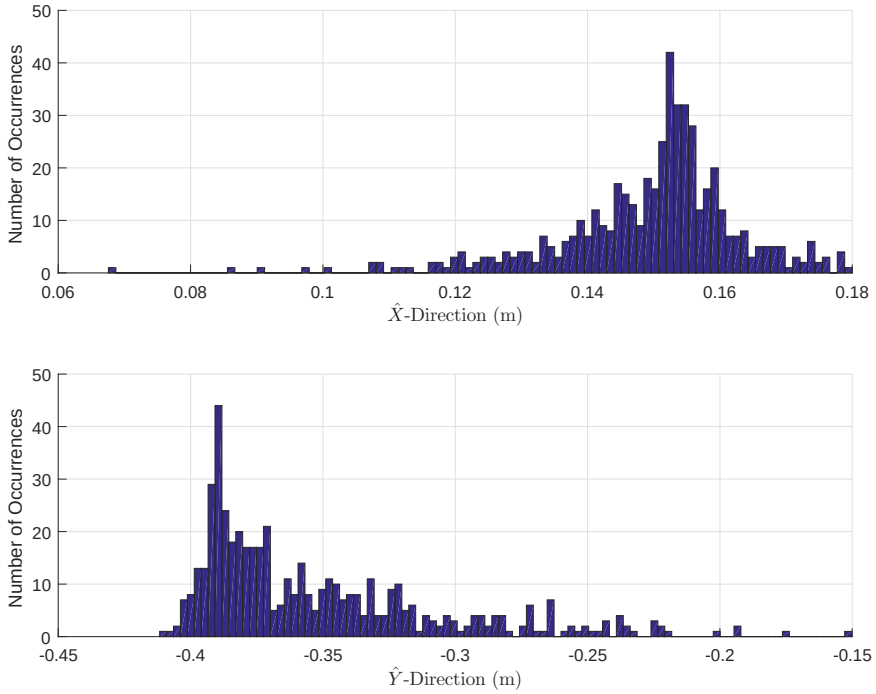

Figure 5.16: Agile Eye camera view MC histogram at $t = 0.4$ sec. Using data at $t = 0.4$ sec, the histogram from the MC analysis of the camera view is not normally distributed.

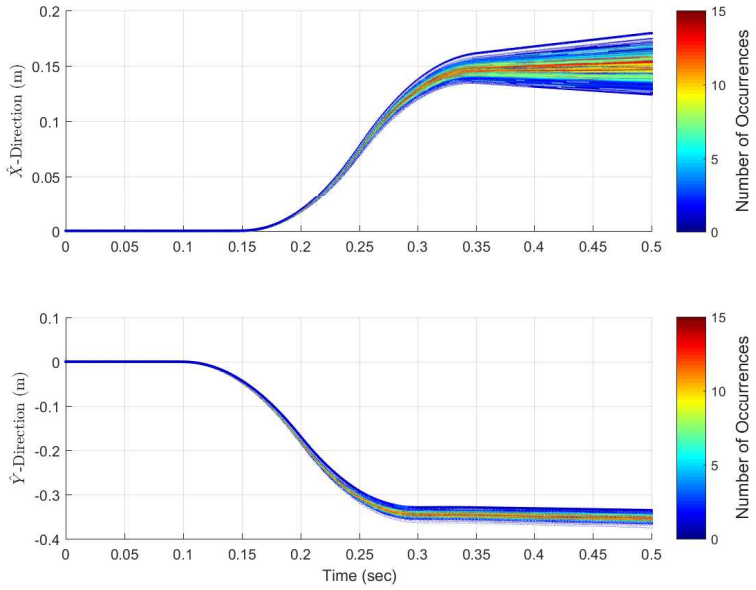

Figure 5.17: Agile Eye camera view PCT time history histogram. The color shows the number of occurrences of at each time. The camera view from the PCT analysis is incorrectly shown to be normally distributed.

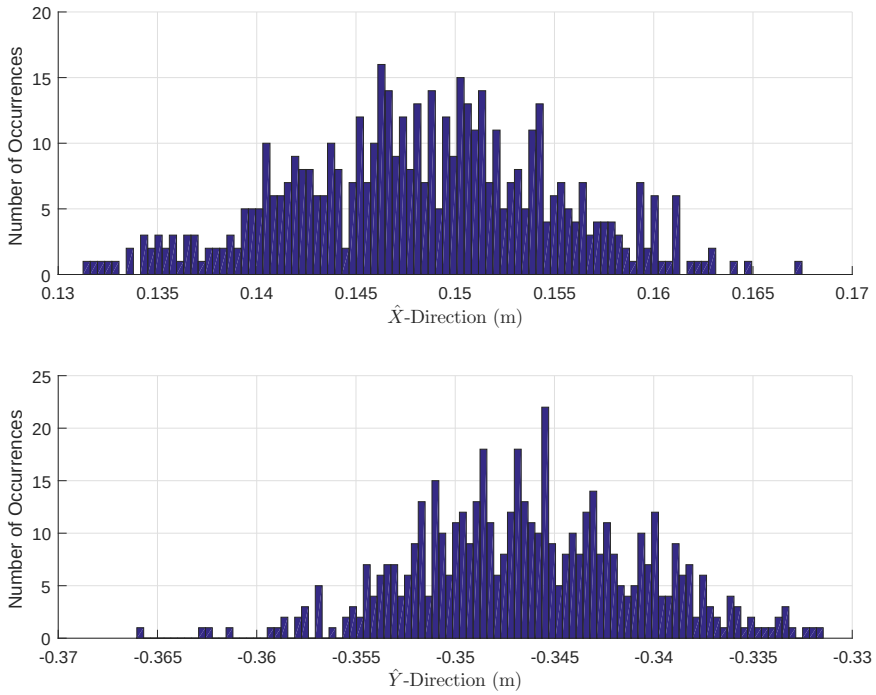

Figure 5.18: Agile Eye camera view PCT histogram at $t = 0.4$ sec. Using data at $t = 0.4$ sec, the histogram from the PCT analysis of the camera view is incorrectly computed to be normally distributed.

not enough terms are carried, then the response of the system can be incorrect. The assumption of the responses being normally distributed is done to keep the size of the system down. Because the responses are not normally distributed, a higher order Hermite polynomial expansion is needed to capture the variation in the responses. Expanding the size of the system is too much for *IDAS* integrator. It is difficult to get *IDAS* to solve a DAE with eighty-four second order differential equations and seventy-eight algebraic position kinematic constraints. If the system is to be expanded to a second order Hermite polynomial, using Eqn. (2.15), the eighty-four becomes 168, and the seventy-eight

becomes 156. Increasing the order of the basis does not even ensure that the PCT expansion captures the response correctly.

5.5 Summary of the Automation Examples

In this chapter, four examples are solved using an MC analysis with DMBoD and PCT with PCMBoD. The slider-crank example shows that the same mean and standard deviations of the states are achieved whether using an MC analysis or PCT; however, PCT is at least ten times faster in this dissertation, when unsupported components are not encountered.

Although the slider-crank example shows the possible benefits of applying PCT to an MBD system, the next three examples show some limitations. The motorcycle traversing a bump shows a limitation with dealing with non-zero length springs in the PCT formulation. The bouncing ball demonstrates difficulties with using PCT on a non-differentiable force such as contact. Finally, a 3D problem, the Agile Eye, shows limitations with the order of the orthogonal polynomial basis set. If the PCT expansion is truncated too much, then an incorrect response variation is calculated.

Two of the examples, the motorcycle traversing a bump and the Agile Eye, show numerical issues as well. The PCT systems get stiffer, causing the integrator to reduce the time step to a prohibitively very small number. Adding PCT to an MBD system makes it

more difficult for the integrator to solve, and computing and exporting the system Jacobian is impractical with the current computing resources. Finding a robust integrator that can solve these types of problems is critical.

Also, without PCMBoD, none of the solutions to the examples is easily achievable. Manually performing the Agile Eye projections with Eqn. (2.19) is incredibly difficult if done by hand, even using the system of two ODEs instead of the maximal set. Quaternions are necessary to not have equations with trigonometric functions, which causes even more complexity in the Agile Eye. Thus, without PCMBoD, the system stiffness and rotational kinematic drivers cannot be investigated.

For PCT to be useful on any generic MBD problem, some additional capability needs to be added to PCMBoD. Additional algorithms can be written to handle special cases such as the square root. Ultimately, one of the main obstacles to overcome is finding or building an appropriate integrator to handle the added complexity that PCT adds to the MBD problem. There is a need for a more robust integrator to handle polynomial truncation. Unless the responses of a system are known to be normally distributed and the number of random variables is low, then an MC analysis is better to use. Finally, if the system contains a non-differential function, such as found in a contact, an MC analysis is the correct method to use.

CHAPTER 6

Conclusion and Future Research

There is a need for a fast and accurate way to perform an uncertainty analysis (UA) in a multi-body dynamics (MBD) system. To understand the response of a system based on the variations that exist in the inputs or system parameters is critical to ensure the system stays within its operational limits or design envelope. MBD systems can contain hundreds of states and have various nonlinearities associated with them. Because of the size of the systems and inherent nonlinearities, the dominant UA method used for an MBD system is a Monte Carlo (MC) analysis. A potentially better UA for MBD systems is polynomial chaos theory (PCT), which can handle nonlinearities. PCT has the potential of decreasing cycle time by embedding the stochastic parameters into the system and solving the resulting larger system once.

6.1 Conclusion

To perform a UA analysis that uses PCT, first, an automation process needs to be created. The deterministic multi-body dynamic (DMBoD) process solves an MBD problem with limited user inputs. DMBoD can support three sets of Euclidean spaces: three-dimensional (3D), two-dimensional (2D) with rotation, and 2D without rotation. Every body in the model has the same number of generalized coordinates depending on

the Euclidean space chosen. All rotational degrees of freedom for a body are represented by quaternions, even in the 2D with rotation Euclidean space. A set of standard holonomic kinematics constraints and joints are used to ensure a consistent set of kinematic constraints is applied to the system. DMBoD takes the inputs entered by the user, and using Maple, derives the differential algebraic equations (DAE) representing the equations of motion using a Constrained Lagrangian method. The equations are converted to state space and exported as Matlab functions containing vectors and matrices for Eqn. (3.39). The user enters the numerical information into Matlab, and then the system is solved. The outputs or system responses of the model are then plotted.

With an automation process created to solve an MBD problem, the next step is to incorporate PCT into the process. However, incorporating PCT into an MBD formulation poses some challenges. The Galerkin projection in Eqn. (2.19) is complex. To address the complexity associated with the Galerkin projection and the application of PCT to an MBD system, several solutions are proposed in this dissertation.

- A trigonometric function, which can be found in all MBD systems with rotations, is a transcendental function, which if used explicitly with a Galerkin projection, yields a complex number which is not able to be used in the solution. To address this issue, quaternions are used to derive the MBD equations of motion for both two-dimensional (2D) and three-dimensional (3D) formulations. Most of the trigonometric functions that would appear traditionally in the MBD equations of

motion are eliminated by using quaternions. Without having to approximate the trigonometric functions, a more exact and efficient solution using quaternions is possible for an MBD system with uncertainty.

- Traditionally, PCT is applied to the final equations of motion of an MBD system. Applying Eqn. (2.19) to acceleration-based terms leads to very complicated calculations. A new method to reduce the complexity of the Galerkin projection, the concept of Variational Work, is derived in this dissertation. The concept of Variational Work applies the Galerkin projection to simpler equations rather than the full set of the equations of motion. Using Variational Work, the Galerkin projection is used on energy functions, which contain velocities, rather than on acceleration functions, which are more complicated. Thus, the application of PCT is embedded into the derivation of the equations of motion, rather than an added step at the end. This application of PCT allows for an easier implementation into an automated process.

- To further simplify the application of the PCT method to an MBD system, an automation process, the polynomial chaos multi-body dynamics (PCMBoD) process, is created. The PCMBoD process is based on the DMBoD process. The PCMBoD process incorporates Variational Work and quaternions into the automation process. With limited user inputs and a standard set of kinematic constraints, the PCT equations of motion are built automatically and solved. The

user does not need to derive manually the equations of motion and perform Galerkin projections. The process is automated fully. The PCMBoD process is capable of analyzing MBD systems with uncertain system responses faster and easier than an MC analysis.

Using these solutions, the application of PCT to a mechanical system is more seamless. There have been previous attempts at applying PCT to an MBD system. Sandu et al. [48, 49] applied PCT to a quarter-car model. Voglewede et al. [56, 57] applied the PCT to an open kinematic chain mechanism. These problems are planar and reduce to a system of ordinary differential equations (ODE). With the use of PCMBoD, all of these problems can be solved quicker and more accurately. No additional simplifications need to be made. There are no Taylor series approximations of trigonometric functions as seen in [57]. Quaternions have eliminated this issue. With the PCMBoD process, the complexity of problems that can be solved has increased. A full 3D mechanism is able to be simulated, such as the Agile Eye, which was not possible before.

In the previous work [48, 49, 56, 57], because the MBD problems reduced to an ODE, there was no issue solving them. The stiffness of the system and its impact on the solution were never discussed. Using PCMBoD on the motorcycle and Agile Eye example, the impact of the system stiffness on the PCT formulation is very noticeable. Without PCMBoD, the impact PCT has on the stiffness of an MBD system would not have come to light.

Using PCMBoD, four example problems are simulated and compared to an MC analysis. Some of these examples are more real-world, less textbook based problems than any problems solved in the literature. These examples illustrate certain types of standard MBD components that are not supported in PCMBoD at this time, which was not known before. These components include common ones, such as a rotational kinematic driver, contact forces and non-zero length springs.

The purpose of this dissertation is to find a new method of analyzing a UA for an MBD system. The PCMBoD process meets the main criteria being sought for a new UA.

- PCMBoD is general and not an ad hoc process.
- PCMBoD is automated with a limited amount of inputs from the user.
- PCMBoD is as fast as an MC analysis, and in most cases much faster, as long as the MBD system contains compatible components.
- PCMBoD quickly and easily post-processes the outputs or system responses.

Using a standard set of generalized coordinates and kinematic constraints made a general automated process with limited user inputs possible. Variational Work allowed PCT to be embedded into the automation process.

Using PCMBoD, PCT is able to replace an MC analysis in most cases. Not all of the standard MBD components are supported by PCMBoD, and there is a possible issue

with the system stiffness. The development of PCMBoD is not finished, but shows great promise for replacing an MC analysis. If the types of distribution of the responses of the system are understood, and the system does not contain any components that are not supported, PCMBoD should be used as the given UA of choice.

6.2 Future Research

To have PCMBoD achieve its full potential and replace an MC analysis, there are additional opportunities for research. The additional opportunities for further research studies fall into two categories: compatible component application and the impact of PCT on the stiffness of a mechanical system.

6.2.1 Polynomial Chaos Theory Compatible Multi-Body Dynamic Components

As shown for the first time in this dissertation, there are certain types of common mechanical components that are unable to be used in the current PCMBoD process due to limitations with the form of the constitutive relationships (e.g., nonlinearities). Specifically, the components discovered in this dissertation that are not compatible with PCMBoD are rotational kinematic drivers, non-zero length springs, and contact forces. While this list is not exhaustive, it shows the breadth of common components that are not compatible with PCMBoD.

Ultimately, finding a way to project these functions in general onto an orthogonal polynomial basis can be the answer to this issue. There are algorithms that have been developed to help with specific types of functions. For rational functions of polynomials, the method in [14] where the rational function is rearranged to be a product of polynomials can be implemented. For square roots, a process based on the method described by [10] where a new variable is introduced that is squared instead of being a square root can be used. For transcendental functions, the Taylor series approach can be used. This approach is already done for trigonometric functions in [57], and a simple algorithm is implemented in [45]. Finally, how to project a discontinuous function onto an orthogonal basis is unknown at this time. A solution is possibly using a discontinuous Galerkin projection method that is common in finite element method [6]. Using the PCMBoD architecture, these additions can be added and the functionality of PCMBoD increased.

Finally, there is an opportunity for putting all of these algorithms together in the current PCMBoD architecture. PCMBoD needs to determine automatically that a function contains one of the special cases and use the appropriate algorithm to project the function onto the orthogonal polynomial basis. Detecting special functions and projecting them is not straightforward for complicated functions. To incorporate this new functionality into PCMBoD, additional functions can be written to handle the detection and projection.

6.2.2 The Impact of Polynomial Chaos Theory on the Stiffness of a Multi-Body Dynamic System

The motorcycle and Agile Eye examples illustrate how the stiffness of a system when PCT is applied to a differential algebraic equation (DAE) representation of an MBD system increases. This work is the first to uncover how the stiffness changes for a PCT MBD system. Understanding how the projection of a system onto an orthogonal polynomial basis affects the stiffness of the resulting DAE is critical. There is an opportunity to address this impact. Stiffness is the reason why a system can run in the DMBoD process, but fails to complete in the PCMBoD process due to an integration time step falling below acceptable tolerances. This research can take two routes: inclusion of the system Jacobian into the solver and/or a custom designed and tuned PCT DAE integrator.

The intent of including the system Jacobian into the solver is to aid the integrator in the solution of stiffer systems. By default, *IDAS* computes the system Jacobian numerically, but is able to take an analytical Jacobian as an input. The obvious approach of using Maple to evaluate Eqn. (5.30) analytically proves to be too time consuming. Perhaps using automatic differentiation inside the Matlab environment, or as an additional step in another software program, can be better. Using automatic differentiation can produce a system Jacobian in a reasonable amount of time allowing more complicated

systems to be solved. Either way, including a better approximation to the system Jacobian can improve the capability of the integrator. Incorporating this functionality into the PCMBoD process ensures a robust application to MBD problems.

The second path is to determine or develop the best class of integrators specifically for a PCT mechanical system DAE. Other DAE integrators are available, such as *DAETS* [41], which may be suitable for this application. If this particular DAE integrator is not applicable, then one could be developed specifically for these types of problems. Also, a different class of DAE integrators can be investigated based on differential variational inequality (DVI) solvers which have benefits when dealing with discontinuities [42]. However, developing a specific DAE integrator is very difficult since most of the DAE integrators classify the type of problems that can be solved by the stiffness of the system. By understanding the impact PCT has on the stiffness of the DAE system, the system could be classified, and an appropriate solver may be able to be developed. The impact of the order of the orthogonal polynomial basis has on stiffness also can be studied. In this dissertation, a first order Hermite polynomial expansion is used, and the systems get stiffer. There is an opportunity understanding how the order of the orthogonal polynomial basis affects the stiffness. Using PCMBoD, the PCT orthogonal polynomial basis order can be expanded easily and aid in the investigation.

Ensuring that the compatible MBD components and the appropriate integrator is developed or chosen for a PCT system, PCMBoD will become more powerful. The goal is

to have any MBD system analyzed with PCT, which has been shown to be more accurate and faster using PCMBoD. PCMBoD in its current state is closer to having a PCT-based UA replace an MC analysis as the default analysis for an MBD system. With additional research, PCMBoD can achieve this goal fully.

REFERNCES

[1] Y. Aminov, "FBD - find the best distribution tool," Matlab Central File Exchange. Retrieved: October 10, 2018. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/36000-fbd-find-the-best-distribution-tool

[2] I. Babuska, R. Tempone, and G. E. Zouraris, "Galerkin finite element approximations of stochastic elliptic partial differential equations," *SIAM Journal on Numerical Analysis*, vol. 42, no. 2, pp. 800–825, 2004.

[3] D. Baraff, "Linear-time dynamics using Lagrange multipliers," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 137–146.

[4] O. A. Bauchau, "Computational schemes for flexible, nonlinear multi-body systems," *Multibody System Dynamics*, vol. 2, no. 2, pp. 169–225, 1998.

[5] F. Caron, "Analyse et conception d'un manipulateur pallèle sphèrique à deux degrès de libertè pour l'orientation d'une camèra," Master's thesis, Universitè Laval, Août 1997.

[6] B. Cockburn, G. E. Karniadakis, and C.-W. Shu, "The development of discontinuous Galerkin methods," in *Discontinuous Galerkin Methods*. Springer, 2000, pp. 3–50.

[7] M. Crosetto and S. Tarantola, "Uncertainty and sensitivity analysis: Tools for GIS-based model implementation," *International Journal of Geographical Information Science*, vol. 15, no. 5, pp. 415–437, 2001.

[8] Dassault Systémes Simulia Corp, "Simulia: Simulation for product, nature & life," accessed: July 23, 2018. [Online]. Available: https://www.3ds.com/products-services/simulia/

[9] L. de robotique, "The agile eye," accessed: July 23, 2018. [Online]. Available: https://robot.gmc.ulaval.ca/en/research/research-thrusts/parallel-mechanisms/the-agile-eye/

[10] B. J. Debusschere, H. N. Najm, P. P. Pébay, O. M. Knio, R. G. Ghanem, and O. P.

Le Maıtre, “Numerical challenges in the use of polynomial chaos representations for stochastic processes,” *SIAM Journal on Scientific Computing*, vol. 26, no. 2, pp. 698–719, 2004.

[11] A. Der Kiureghian and J.-B. Ke, “The stochastic finite element method in structural reliability,” in *Stochastic Structural Mechanics*. Springer, 1987, pp. 84–109.

[12] U. M. Diwekar and S. Ulas, *Sampling Techniques*, online ed. John Wiley & Sons, Inc., 2000.

[13] D. J. Downing, R. Gardner, and F. Hoffman, “An examination of response-surface methodologies for uncertainty analysis in assessment models,” *Technometrics*, vol. 27, no. 2, pp. 151–163, 1985.

[14] J. Fisher and R. Bhattacharya, “Optimal trajectory generation with probabilistic system uncertainty using polynomial chaos,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 133, no. 1, p. 014501, 2011.

[15] C. W. Gear, “Differential-algebraic equation index transformations,” *SIAM Journal on Scientific and Statistical Computing*, vol. 9, no. 1, pp. 39–47, 1988.

[16] C. Gear, B. Leimkuhler, and G. Gupta, “Automatic integration of Euler-Lagrange equations with constraints,” *Journal of Computational and Applied Mathematics*, vol. 12–13, pp. 77 – 90, 1985.

[17] J. Ginsberg, *Engineering Dynamics*. New York: Cambridge University Press, 2008.

[18] D. T. Greenwood, *Advanced Dynamics*. New York: Cambridge University Press, 2006.

[19] J. H. Halton, “A retrospective and prospective survey of the Monte Carlo method,” *SIAM Rev.*, vol. 12, no. 1, pp. 1–63, Jan. 1970.

[20] M. E. Harr, “Probabilistic estimates for multivariate analyses,” *Applied Mathematical Modelling*, vol. 13, no. 5, pp. 313–318, 1989.

[21] R. L. Harrison, “Introduction to Monte Carlo simulation,” *AIP Conference Proceedings*, vol. 1204, pp. 17–21, January 2010.

[22] E. J. Haug, *Computer Aided Kinematics and Dynamics of Mechanical Systems. Vol.*

1: Basic Methods. Needham Heights, MA: Allyn & Bacon, Inc., 1989.

[23] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, "SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers," *ACM Trans. Math. Softw.*, vol. 31, no. 3, pp. 363–396, Sep. 2005.

[24] A. C. Hindmarsh, R. Serban, and A. Collier, "User documentation for IDA v2. 9.0 (SUNDIALS v2. 7.0)," 2016.

[25] F. Hoffman, J. Hammonds, and S. M. Bartell, "An introductory guide to uncertainty analysis in environmental and health risk assessment. environmental restoration program," Oak Ridge National Laboratory, Oak Ridge, TN, Tech. Rep. ES/ER/TM-35/Rl, 1994.

[26] T. R. Kane and D. A. Levinson, *Dynamics, Theory and Applications*. New York: McGraw Hill, 1985.

[27] G. Kewlani, J. Crawford, and K. Iagnemma, "A polynomial chaos approach to the analysis of vehicle dynamics under uncertainty," *Vehicle System Dynamics*, vol. 50, no. 5, pp. 749–774, 2012.

[28] J. P. C. Kleijnen, "Sensitivity analysis and related analyses: A review of some statistical techniques," *Journal of Statistical Computation and Simulation*, vol. 57, pp. 111–142, 1997.

[29] B. Kriegesmann, "Probabilistic design of thin-walled fiber composite structures," Ph.D. dissertation, Mitteilungen des Instituts für Statik und Dynamik der Leibniz Universität Hannover, April 2012.

[30] S. H. Lee and W. Chen, "A comparative study of uncertainty propagation methods for black-box-type problems," *Structural and Multidisciplinary Optimization*, vol. 37, no. 3, pp. 239–253, 2009.

[31] W. K. Liu, T. Belytschko, and A. Mani, "Probabilistic finite elements for nonlinear structural dynamics," *Computer Methods in Applied Mechanics and Engineering*, vol. 56, no. 1, pp. 61–81, 1986.

[32] Mathworks, "Simscape Multibody," accessed: July 23, 2018. [Online]. Available: https://www.mathworks.com/products/simmechanics.html

[33] Mathworks, "Solve differential algebraic equations (DAEs)," https://www.mathworks.com/help/matlab/math/solve-differential-algebraic-equations-daes.html, accessed: January 22, 2017.

[34] H. G. Matthies, C. E. Brenner, C. G. Bucher, and C. G. Soares, "Uncertainties in probabilistic numerical analysis of structures and solids-stochastic finite elements," *Structural Safety*, vol. 19, no. 3, pp. 283 – 336, 1997.

[35] Motion Port, "RecurDyn multibody dynamics simulation," accessed: July 23, 2018. [Online]. Available: http://www.motionport.com/index.aspx?page=RecurDyn

[36] MSC Software Corporation, "Adams the multibody dynamics simulation solution," accessed: July 23, 2018. [Online]. Available: http://www.mscsoftware.com/product/adams

[37] MSC.Adams, "Using the Adams/View function builder," 2014.

[38] MSC.Adams, "Welcome to the C++ version of Adams/Solver," 2014.

[39] R. M. Murray, S. S. Sastry, and L. Zexiang, *A Mathematical Introduction to Robotic Manipulation*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 1994.

[40] R. H. Myers, A. I. Khuri, and W. H. Carter, "Response surface methodology: 1966–l988," *Technometrics*, vol. 31, no. 2, pp. 137–157, 1989.

[41] N. Nedialkov and J. Pryce, "DAETS: Differential-algebraic equations by Taylor series," accessed: August 26, 2018. [Online]. Available: http://www.cas.mcmaster.ca/~nedialk/daets/

[42] J.-S. Pang and D. E. Stewart, "Differential variational inequalities," *Mathematical Programming*, vol. 113, no. 2, pp. 345–424, 2008.

[43] R. Pulch, "Polynomial chaos for linear differential algebraic equations with random parameters," *International Journal for Uncertainty Quantification*, vol. 1, no. 3, pp. 223–240, 2011.

[44] D. G. Robinson, "A survey of probabilistic methods used in reliability, risk and uncertainty analysis: Analytical techniques 1," Sandia National Laboratories,

Albuquerque, NM, Tech. Rep. SAND98-1189, June 1998.

[45] P. S. Ryan, S. Baxter, and P. A. Voglewede, "Variational analysis of a two link slider-crank mechanism using polynomial chaos theory," in *Proceedings of the 2017 ASME IDETC*, no. DETC2017-67328, 2017.

[46] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global Sensitivity Analysis. The Primer*. John Wiley & Sons, Ltd, 2008.

[47] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto, *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models*. New York, NY, USA: Halsted Press, 2004.

[48] A. Sandu, C. Sandu, and M. Ahmadian, "Modeling multibody systems with uncertainties. Part I: Theoretical and computational aspects," *Multibody System Dynamics*, vol. 15, no. 4, pp. 369–391, 2006.

[49] C. Sandu, A. Sandu, and M. Ahmadian, "Modeling multibody systems with uncertainties. Part II: Numerical applications," *Multibody System Dynamics*, vol. 15, no. 3, pp. 241–262, 2006.

[50] R. Sharp, S. Evangelou, and D. J. Limebeer, "Advances in the modelling of motorcycle dynamics," *Multibody System Dynamics*, vol. 12, no. 3, pp. 251–283, 2004.

[51] Siemens PLM Software, "LMS Virtual.Lab Motion," accessed: July 23, 2018. [Online]. Available: https://www.plm.automation.siemens.com/fr/products/lms/virtual-lab/motion

[52] A. H. C. Smith, "Robust and optimal control using polynomial chaos theory," Ph.D. dissertation, University of South Carolina, 2007.

[53] G. Stefanou, "The stochastic finite element method: past, present and future," *Computer Methods in Applied Mechanics and Engineering*, vol. 198, no. 9, pp. 1031–1051, 2009.

[54] B. Sudret and A. Der Kiureghian, *Stochastic finite element methods and reliability: a state-of-the-art report*. Department of Civil and Environmental Engineering,

University of California Berkeley, CA, 2000.

[55] T. Uchida and J. McPhee, "Triangularizing kinematic constraint equations using gröbner bases for real-time dynamic simulation," *Multibody System Dynamics*, vol. 25, no. 3, pp. 335–356, 2011.

[56] P. Voglewede, A. H. C. Smith, and A. Monti, "Dynamic performance of a SCARA robot manipulator with uncertainty using polynomial chaos theory," *IEEE Transactions on Robotics*, vol. 25, no. 1, pp. 206–210, Feb 2009.

[57] P. A. Voglewede and A. Monti, "Variation analysis of a two link planar manipulator using polynomial chaos theory," in *Proceedings of the 2006 ASME IDETC*, vol. Volume 2: 30th Annual Mechanisms and Robotics Conference, Parts A and B, no. 10.1115/DETC2006-99170, 2006.

[58] R. W. Walters and L. Huyse, "Uncertainty analysis for fluid mechanics with applications," NASA, Hampton, VA, Tech. Rep. No. ICASE-2002-1, February 2002.

[59] C. Wang, "Parametric uncertainty analysis for complex engineering systems," Ph.D. dissertation, Massachusetts Institute of Technology, 1999.

[60] N. Wiener, "The homogeneous chaos," *American Journal of Mathematics*, vol. 60, no. 4, pp. 897–936, 1938.

[61] D. Xiu, "Fast numerical methods for stochastic computations: A review," *Communications in Computational Physics*, vol. 5, no. 2-4, pp. 242–272, 2009.

[62] D. Xiu and G. E. Karniadakis, "The Wiener-Askey polynomial chaos for stochastic differential equations," *SIAM Journal on Scientific Computing*, vol. 24, no. 2, pp. 619–644, 2002.

[63] D. Xiu, *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton, New Jersey: Princeton University Press, 2010.

APPENDIX A

Maple Functions

A series of functions built for Maple that aid in the generation of a deterministic or polynomial chaos theory (PCT) expanded multi-body dynamics (MBD) system.

A.1 Kinematic Constraint Functions

A.1.1 Three Dimensional

Constraints

- *Dot1Constraint* - Builds dot-1 constraint
- *Dot2Constraint* - Builds dot-2 constraint
- *Parallel1Constraint* - Builds parallel-1 constraints
- *Parallel2Constraint* - Builds parallel-2 constraints
- *SphericalConstraint* - Builds spherical constraints

Joints

- *CylindricalJoint* - Builds constraints for a cylindrical joint

- *FixedJoint* - Builds constraints for a fixed joint
- *PrismaticJoint* - Builds constraints for a prismatic joint
- *RevoluteJoint* - Builds constraints for a revolute joint
- *SphericalJoint* - Builds constraints for a spherical joint
- *SphericalSpherical* - Builds fixed distance constraint
- *UniversalJoint* - Builds constraints for a universal joint

A.1.2 Two Dimensional

Constraints

- *Dot1Constraint2D* - Builds dot-1 constraint
- *Dot2Constraint2D* - Builds dot-2 constraint

Joints

- *FixedJoint2D* - Builds constraints for a fixed joint
- *PrismaticJoint2D* - Builds constraints for a prismatic joint
- *RevoluteJoint2D* - Builds constraints for a revolute joint
- *RevoluteRevolute2D* - Builds fixed distance constraint

A.2 Kinematic Functions

A.2.1 Three Dimensional

- *AngVelquat* - Builds the body-fixed angular velocity from quaternion rates
- *Aquat* - Builds a rotation matrix from a quaternion vector
- *Gquat* - Builds matrix relating body-fixed angular velocity to quaternion rates
- *Rx* - Builds a rotation matrix about an x-axis
- *Ry* - Builds a rotation matrix about an y-axis
- *Rz* - Builds a rotation matrix about an z-axis
- *SkewSymmetric* - Builds a skew-symmetric matrix of a vector

A.2.2 Two Dimensional

- *AngVelquat2D* - Builds the body-fixed angular velocity from quaternion rates
- *Aquat2D* - Builds a two dimensional rotation matrix from a quaternion vector
- *Gquat2D* - Builds matrix relating body-fixed angular velocity to quaternion rates
- *Rz2d* Builds a two dimensional rotation matrix

A.3 Matlab Functions

- *MatlabFunctionIDAS* - Exports a matrix or vector as a Matlab function to be used by the integrator IDAS
- *MatlabFunctionIDASO* - Exports a only non-zero components of a matrix or vector as a Matlab function to be used by the integrator IDAS
- *MatlabFunctionIDASOptimized* - Exports a vector as a Matlab function with Maple "optimized" flag to be used by the integrator IDAS
- *MatlabFunctionIDASParameters* - Exports the parameters as Matlab function to be user inputted in Matlab and to be used by the integrator IDAS
- *StringFix* - Fixes Maple's variable name limitations when export Matlab Code

A.4 Multi-Body Dynamics Functions

- *AppliedGenForces* - Calculate the applied generalized force
- *DelEDelq* - Calculates the derivative of energy with respect to a generalized coordinate
- *ConstraintJacobian* - Calculate the constraint Jacobian
- *DiffState* - Calculates the derivative with respect to a state or generalized coordinate
- *ElimDepOnT* - Eliminates functions of time in variables

- *FrcMom2GenFrc* - Calculates the generalized force or moment
- *GammaVec* - Calculates the gamma vector for acceleration constraints
- *GeneralizedCoordinates* - Builds the generalized coordinate vector
- *KineticEnergyMatrix* - Calculates the kinetic energy of the system
- *LagrangeMultipliers* - Builds the Lagrange Multipliers vector
- *LagrangeT* - Calculates the mass matrix and nonlinear acceleration inertial terms from the kinetic energy
- *PEGenFroces* - Calculates the conservative forces from the potential energy
- *Phit* - Calculate the explicit time derivative of position constraints
- *Phitt* - Calculate the second explicit time derivative of position constraints
- *PotentialEnergy* - Calculates the potential energy of the system
- *QuaternionConstraints* - Calculates unit quaternion constraint and adds it to the position constraints
- *VarDiffT* - Calculates the derivative of an equation with respect to time
- *Velocity2VirtualDisp* - Replaces velocities with virtual displacements

A.5 Multi-Body Dynamics State Space

- *ConServForceSS* - Moves the conservative forces to state space

- *GammaSS* - Moves the constraint equations to state space
- *GenForceSS* - Moves the generalized forces to state space
- *MassMatrixSS* - Moves the mass matrix to state space
- *NLAccSS* - Moves the nonlinear acceleration inertial vector to state space
- *PhiSS* - Moves the constraint equations to state space
- *PhiqSS* - Moves the constraint Jacobian to state space
- *RequestSS* - Moves the requests or outputs to state space and cleans up the PCT variables
- *SSIndx3SI2* - Builds the SI2 state space system
- *Vel2PosDot* - Builds the position states derivative relationship to velocity states

A.6 Polynomial Chaos Theory Functions

- *GeneralizedCoordinatesPCT* - Expands the velocity and acceleration generalized coordinates along the orthogonal polynomial basis
- *GalerkinProjection* - Performs the Galerkin projection
- *GalerkinProjectionEnergy* - Performs the Variational Energy projection
- *GenPCTForce* - Performs the Variational Principle of Virtual Work
- *HermitianPolynomial* - Hermitian polynomial expansion

- *InnerProduct* - Calculates the coefficients of the random variable along the orthogonal polynomial basis
- *PCTOrder* - Determines the order of the polynomial expansion
- *PCTParameters* - Builds the PCT variables and Galerkin projection Weighting function
- *PCTQExpansion* - Expands the generalized coordinates and Lagrange Multipliers along the orthogonal polynomial basis
- *PCTVariableExpansion* - Expands a random variable in a given orthogonal polynomial basis set. Only Hermitian polynomials are supported at this time.

APPENDIX B

Matlab Functions

A series of functions were built for Matlab to solve stabilized index-2 (SI2) differential algebraic equations (DAE) using the *IDAS* integrator [23].

B.1 Monte Carlo Analysis Functions

- *GetRandomVariables* - Builds sample population for Monte Carlo analysis

B.2 Polynomial Chaos Theory Functions

- *BuildPCTDist* - Builds sample population for polynomial chaos theory variables to evaluate outputs

B.3 Simulation Functions

- *AccelLagrangeICs* - Calculates a consistent set of acceleration and Lagrange Multiplier initial conditions (IC)
- *Assembly* - Calculates a consistent set of ICs

- *AssemblyNoCheck* - Calculates a consistent set of ICs without checking final residual
- *DAEI3SI2* - Calculates the residual for an SI2 DAE formulation
- *PositionICs* - Calculates a consistent set of position ICs using Newton-Raphson
- *VelocityICs* - Calculates a consistent set of velocity ICs using Newton-Raphson

APPENDIX C

Motorcycle Numerical Information

The numerical parameters for the Motorcycle Example in Section 5.2 are defined in this Appendix. This information comes from Sharp [50].

- The point locations for the motorcycle are in Table C.1.

Table C.1: Motorcycle point locations [50]

Point	X (m)	Y (m)
p_2	1.1730	0.7490
p_4	1.3420	0.2820
p_6	1.4100	0.2820
p_7	0.0000	0.2970
p_{11}	0.5490	0.3608
p_{13}	0.4870	0.4888
p_{19}	0.5390	0.1878
p_{20}	0.4946	0.1522
p_{21}	0.4443	0.1782
p_{22}	0.3722	0.2748

- The mass properties for the motorcycle are in Table C.2.

Table C.2: Motorcycle mass properties [50]

Body	$p_{cm_{ix}}$ (m)	$p_{cm_{iy}}$ (m)	Mass (kg)	Inertia (kg-m^2)
1	0.6334	0.5855	198.81	37.7614
2	1.1640	0.7700	9.9900	1.5840
3	1.3930	0.2979	19.1500	0.5011
4	0.4946	0.2578	26.6000	1.1439
5	0.5490	0.1522	0.001	0.001

- The system parameters are [50]
  - The rake angle, $\epsilon = 0.4412$ radians,
  - The front stiffness, $k_{ff} = 25000$ N/m,
  - The front damping, $C_{ff} = 2134$ N-sec/m,
  - The rear shock free length, $L_{sa} = 0.3435$ m,
  - The front stiffness, $k_{sa} = 58570$ N/m, and
  - The front damping, $C_{sa} = 11650$ N-sec/m.
- The random variables parameters are
  - The mean of front shock stiffness, $\bar{k_{ff}} = k_{ff}$ N/m,
  - The standard deviation of front shock stiffness, $\sigma_{k_{ff}} = 833.3$ N/m,
  - The mean of rear shock stiffness, $\bar{k_{sa}} = k_{sa}$ N/m, and
  - The standard deviation of rear shock stiffness, $\sigma_{k_{sa}} = 1.952 \times 10^3$ N/m,

- The speed and bump parameters are
  - The speed of the motorcycle, $v_{odom} = 4.4704$ m/sec,
  - The length of the bump, $L_{bump} = 0.3048$ m,
  - The length of the bump, $h_{bump} = 0.0508$ m, and
  - The wheel length, $L_{wheel} = 1.4100$ m.

APPENDIX D

Agile Eye Numerical Information

The numerical parameters for the Agile Eye Example in Section 5.4 are defined in this Appendix. This information comes from Caron [5].

- The center of mass locations for the Agile Eye in Table D.1.

Table D.1: Agile Eye center of mass location [5]

Body	$p_{cm_{ix}}$ (m)	$p_{cm_{iy}}$ (m)	$p_{cm_{iz}}$ (m)
1	0.0000	-4.8700×10^{-4}	-0.0121
2	0.0195	0.0000	0.0000
3	0.0000	0.0000	-0.0375
4	-0.0169	0.0000	-0.0267

- The mass and inertia for the Agile Eye in Table D.2, Table D.3 and Table D.4.

Table D.2: Agile Eye mass [5]

Body	Mass (kg)
1	0.0349
2	0.0000
3	0.0154
4	0.0094

Table D.3: Agile Eye moments of inertia [5]

Body	J_{xx} (kg-m^2)	J_{yy} (kg-m^2)	J_{zz} (kg-m^2)
1	1.3413×10^{-5}	1.3830×10^{-5}	4.5815×10^{-6}
2	3.4040×10^{-5}	2.0830×10^{-5}	5.4400×10^{-5}
3	1.7550×10^{-5}	6.2820×10^{-6}	1.1640×10^{-5}
4	1.1930×10^{-5}	1.5360×10^{-5}	3.6500×10^{-6}

Table D.4: Agile Eye products of inertia [5]

Body	J_{xy} (kg-m^2)	J_{xz} (kg-m^2)	J_{yz} (kg-m^2)
1	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000
3	0.0000	0.0000	0.0000
4	0.0000	2.4930×10^{-6}	0.0000

- The system parameters are [5]
  - $L_1 = 0.06508$ m,
  - $L_2 = 0.06508$ m, and
  - $L_3 = 0.506$ m.
- The applied moment parameters are
  - $M_x = 0.001$ N-m,
  - $M_y = 0.0015$ N-m, and
  - $tol = 0.01$ sec.

- The random variables parameters are
  - The mean of body 1 x-component center of mass, $\bar{x_{cm1}} = x_{cm_1}$ m,
  - The standard deviation of body 1 x-component center of mass, $\sigma_{x_{cm1}} = 0.01$ m,
  - The mean of body 1 y-component center of mass, $\bar{y_{cm1}} = y_{cm_1}$ m, and
  - The standard deviation of body 1 x-component center of mass, $\sigma_{y_{cm1}} = 0.01$ m.